Jaume Tello Viñas

DETECCIÓ INTEL·LIGENT D'ABOCADORS IL·LEGALS: APLICACIÓ MÒBIL AMB IA PER AL REGISTRE I MONITORATGE

TREBALL DE FI DE GRAU

dirigit per Usama Benabdelkrim Zakan

Grau d'Enginyeria Informàtica

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2025

Agraïments

En finalitzar aquest Treball de Fi de Grau, vull deixar constància, amb humilitat i gratitud, de totes les persones i institucions que l'han fet possible. Aquestes línies reconeixen el vostre suport, els consells i la paciència que han convertit AbocadorsBP en una realitat.

A la Mireia, la meva parella, per ser-hi sempre. Pel suport emocional, per acompanyar-me en els moments exigents i per implicar-s'hi de debò, ajudant-me, provant funcionalitats i donant-me calma i empenta quan més ho he necessitat no només al llarg d'aquest projecte sinó al llarg de la nostra vida.

Als meus pares, Toni i Neus, per recolzar-me incondicionalment i per proposar-me el projecte que ha acabat donant forma a aquest TFG. Gràcies per la confiança i per ensenyar-me a perseverar. Gràcies pels valors que m'heu inculcat, perquè aquests són la base de projectes com el que aquí es presenta.

Als meus germans, que sense saber-ho em donen suport, al Pere, per la paciència quan ocupo l'ordinador durant hores infinites sense deixar-lo jugar; i al Pau, per recomanar-me a persones a qui fer-los aplicacions i pel suport constant. Sou una part important d'aquest camí.

Al meu tutor, Usama Benabdelkrim Zakan, perquè més enllà de donar-me base com a professor al llarg de la carrera, m'ha orientat amb criteri en la presa de decisions d'aquest projecte. Tinc una gran admiració per la seva dedicació i disponibilitat.

A la resta de la família i amics, per les idees de millora, pel temps dedicat a testejar l'aplicació i per l'honestedat dels vostres comentaris i el vostre suport. En especial, al Kike i al Toni, per aguantar-me quan juguem o les meves desaparicions durant setmanes, els moments de descans eren necessaris, visca TDP.

A l'avia Rosa Maria i als que ja no hi són, la tieta Montse, que sempre estava tan orgullosa que fos tan estudiós; a l'avi Antonio, que em repetia que em centrés en els estudis; a l'avi Pere i a la iaia Anita. Aquest treball també és vostre.

Finalment, al professorat de la Universitat Rovira i Virgili, per l'exigència, la qualitat docent i els valors que m'emporto. Gràcies per ajudar-me a convertir una idea en una eina real al servei del territori.

Tanco aquests agraïments amb la certesa que aquest projecte continuarà creixent amb el vostre esperit de col·laboració. A tots els qui m'heu acompanyat en el camí: moltes gràcies de tot cor.

Resum.

Els abocadors il·legals degraden el medi i requereixen actuacions lentes i costoses. Des d'una motivació personal, aquest TFG presenta AbocadorsBP, una solució per agilitzar la detecció i implicar la ciutadania. L'arquitectura combina una app mòbil multi plataforma (React Native + Expo), una API amb FastAPI i una base de dades PostgreSQL. La denúncia es fa fotografiant l'abocador, geolocalitzant-lo automàticament i enviant-lo a una base central. Un model d'IA (YOLOv11) assisteix en la detecció i classificació de residus; els casos s'exploren al mapa (Leaflet/OpenStreetMap) i es disposa d'estadístiques i rols d'usuari/administrador. Els resultats mostren una solució operativa, amb detecció automàtica viable (mAP@50–95 55% i millorable amb més dades), temps de resposta adequats i un flux complet de registre i gestió. El projecte seguirà en desenvolupament posterior a la publicació i considera el contacte amb administracions públiques, potenciant una resposta més proactiva i un impacte social i ambiental positiu.

Resumen.

Los vertederos ilegales degradan el entorno y exigen respuestas lentas y costosas. A partir de una motivación personal, este TFG presenta AbocadorsBP, una solución para agilizar la detección e implicar a la ciudadanía. La arquitectura integra una app móvil (React Native + Expo), una API con FastAPI y base de datos PostgreSQL. La denuncia se realiza fotografiando el vertedero, geolocalizándolo automáticamente y enviándolo a una base central. Un modelo de IA (YOLOv11) asiste en la detección y clasificación de residuos; los casos se exploran en un mapa (Leaflet/OpenStreetMap) con estadísticas y gestión por roles. Los resultados muestran una solución operativa, con detección automática viable (mAP@50–95 55% y mejorable con más datos), tiempos de respuesta adecuados y un flujo completo de registro y gestión. El proyecto seguirá en desarrollo posterior a su publicación y considera el contacto con administraciones públicas, potenciando una respuesta más proactiva y un impacto social y ambiental positivo.

Abstract.

Illegal dumping harms ecosystems and public health, while traditional responses are slow and resource-intensive. Based on a personal motivation, this thesis presents AbocadorsBP, a citizen-driven solution. The system combines a cross-platform mobile app (React Native + Expo), a FastAPI backend, and PostgreSQL. Users report sites by taking a photo, which is auto-geolocated and stored centrally. An AI model (YOLOv11) assists waste detection and classification; reports are explored on interactive maps (Leaflet/OpenStreetMap) with statistics and role-based management. Results show an operational solution with viable automated detection (55% mAP@50–95, improvable with more data), acceptable response times, and an end-to-end reporting/management workflow. The project will continue to develop after publication and considers contact with public administrations, promoting a more proactive response and a positive social and environmental impact.

Índex

Índex de taules

Índex de figures

1 Introducció

El canvi climàtic és un repte global complex i multifactorial, que exigeix la participació coordinada de tots els actors socials com individus, empreses, governs i sectors industrials. Segons l'**IPCC**[1] (2022), les activitats humanes han provocat un escalfament global d'1,1°C respecte a nivells preindustrials, amb impactes irreversibles en els ecosistemes [1][2][3]. Si aquestes activitats no canvien existeix el risc de que aquest escalfament arribi a ser d'entre 2.6-3.1°C [4][5][6].

Tot i que el pes de la mitigació recau principalment en polítiques públiques i corporatives (els països del G20 responsables del **80% de les emissions globals**, segons l'UNEP[2] [4]), l'acció individual també és rellevant com a motor de canvi social i pressió ciutadana. La ciutadania pot contribuir a través de decisions quotidianes més sostenibles i de la implicació activa en la preservació del medi ambient. En aquest sentit, una de les manifestacions visibles del maltractament ambiental és la proliferació d'abocadors il·legals, especialment en zones rurals o periurbanes, on el control administratiu és menys intens. Aquests abocadors no només dificulten el correcte desenvolupament de la fauna i flora a les zones afectades sinó que introdueixen una contaminació visual radical.

En aquest context de necessitat d'acció col·lectiva, la tecnologia digital emergeix com una eina fonamental per facilitar la participació ciutadana. Les aplicacions mòbils, la intel·ligència artificial i els sistemes d'informació geogràfica ofereixen oportunitats inèdites per proporcionar accés a la monitorització ambiental i amplificar l'impacte de les accions individuals.

Aquest treball de final de grau neix d'una motivació personal profundament arrelada. Des de ben petit, he crescut en un entorn familiar amb una forta consciència ecològica. El meu pare, gran aficionat al ciclisme i compromès amb la conservació del territori, fa anys que recorre els camins del Baix Penedès i comarques dels voltants documentant manualment abocadors il·legals que troba durant les seves sortides. Malgrat els esforços que ha dut a terme durant més de quatre anys (entrevistes als mitjans, conferències en centres educatius, i la creació de contingut per a xarxes socials per sensibilitzar la població), aquesta tasca de documentació ha continuat recaient exclusivament sobre ell.

És per aquest motiu que va començar a demanar-me el desenvolupament d'una eina tecnològica que permetés agilitzar el procés i fer-lo col·laboratiu. Aquesta petició va suposar per a mi una oportunitat única per posar en pràctica coneixements apresos durant el grau, aprofundir en àmbits tecnològics que m'interessen especialment, com la IA[3], i alhora contribuir activament a una causa social i mediambiental que em toca de prop. Com a valor afegit, aquest projecte em permet introduir-me en àmbits de la informàtica que no desperten tant la meva curiositat però que poden ser-me molt útils en un futur, com el desenvolupament mòbil.

[1] IPCC: Intergovernmental Panel on Climate Change

[2] UNEP: United Nations Environment Programme

[3] IA: Intel·ligència Artificial

Des de fa temps tenia la voluntat d'incorporar la intel·ligència artificial al meu treball de final de grau, especialment en l'àmbit de la visió per computador, per tal d'explorar com aquesta tecnologia pot tenir un impacte real i positiu en problemes del món físic. Inicialment es va considerar el desenvolupament d'una aplicació mòbil destinada a fomentar el reciclatge. La funcionalitat principal consistia a permetre a l'usuari fer una fotografia d'un residu domèstic (com ara un envàs, un objecte de plàstic, una pila o un Tetra Brick) i que l'aplicació, mitjançant un model de classificació d'imatges basat en intel·ligència artificial, indiqués a quin contenidor corresponia el seu correcte reciclatge. A partir d'aquesta proposta inicial i seguint la motivació del meu pare, es va acabar arribant a la proposta actual: **AbocadorsBP**, una aplicació mòbil que integra detecció automàtica de residus mitjançant intel·ligència artificial, geolocalització i participació ciutadana per crear una eina col·laborativa de monitorització territorial.

Aquest document presenta el disseny, desenvolupament i validació d'aquesta solució tecnològica, analitzant tant els reptes tècnics com l'impacte social de la seva implementació.

1.1 Declaració de titularitat dels drets d'autor

En virtut del que estableix la *Normativa sobre propietat industrial i intel·lectual de la Universitat Rovira i Virgili* [7] (aprovada el 30 d'abril de 2009 i modificades el 22 de febrer de 2018), es fa constar que aquest Treball de Fi de Grau (TFG), incloent-hi el desenvolupament de l'aplicació, el disseny funcional i tècnic, el codi font i la documentació associada, ha estat concebut i desenvolupat íntegrament per l'estudiant Jaume Tello Viñas, sense que existeixi cap encàrrec específic ni projecte de recerca institucional de la URV o de tercers que en condicioni l'autoria.

D'acord amb l'article 5.2 de la normativa esmentada, l'autoria i la titularitat dels drets d'explotació corresponen a l'estudiant, com a creador d'una obra desenvolupada en el marc de les seves activitats acadèmiques. Per tant, qualsevol ús, reproducció, distribució o explotació comercial del programari resultant correspon exclusivament a l'autor, sens perjudici del reconeixement obligatori a la Universitat Rovira i Virgili en qualitat d'entitat on s'ha presentat i avaluat aquest treball.

L'autor autoritza la URV únicament a la consulta, reproducció i difusió del TFG amb finalitats acadèmiques i de recerca, dins dels canals institucionals de la universitat, sense que això suposi cessió dels drets d'explotació.

1.2 Política de Privadesa

Aquest projecte disposa d'una política de privadesa específica d'AbocadorsBP, on es descriu el tractament de dades (identificació, imatges i geolocalització), les finalitats, les mesures de seguretat i els drets de les persones usuàries. Darrera actualització: 11/08/2025.

2 Descripció General del Projecte

El projecte que aquí es presenta consisteix en el disseny i desenvolupament d'una aplicació mòbil per a la detecció, registre i monitoratge d'abocadors il·legals mitjançant intel·ligència artificial. L'aplicació permet als usuaris capturar imatges d'abocadors, afegir-ne la ubicació mitjançant geolocalització i, mitjançant un model de detecció i classificació d'objectes basat en la tecnologia **YOLO[4]v11** [8], reconèixer quin tipus de deixalles es troben en aquest abocador, per tal de que l'usuari només hagi d'encarregar-se de fer la fotografia. D'aquesta manera, no només es redueix la càrrega per a l'usuari a l'hora de registrar un abocador, sinó que també s'aconsegueix estructurar millor les dades recollides per a futurs anàlisis i accions. Les dades es centralitzen en una base de dades comuna accessible des d'un mapa interactiu, amb l'objectiu de fomentar la participació ciutadana, optimitzar la detecció i facilitar l'acció institucional.

Aquesta aplicació s'estructura com una solució tecnològica completa que aborda una problemàtica mediambiental de gran rellevància social: la gestió dels abocadors il·legals a la comarca del Baix Penedès. Cal contextualitzar que els abocadors il·legals constitueixen un problema persistent que afecta tant el paisatge com la salut pública, i la seva detecció tradicional requereix recursos humans considerables i processos administratius lents. Aquest projecte digitalitza i automatitza gran part d'aquest procés, creant un pont entre la ciutadania i les administracions locals tal com es promou en plataformes de ciència ciutadana per a temes ambientals [9].

L'aplicació resulta adequada per a diferents tipus d'usuaris. D'una banda, ciutadans conscienciats que vulguin reportar abocadors il·legals de manera senzilla i eficaç, sense necessitat de coneixements tècnics avançats. De l'altra, administracions locals i personal tècnic que necessitin gestionar aquestes denúncies de forma organitzada i eficient.

Pel que fa a l'arquitectura del sistema, el projecte es divideix en tres components principals clarament diferenciats. En primer lloc, **AbocadorsBPApp**, una aplicació mòbil desenvolupada amb React Native i Expo [10] que constitueix la interfície principal per als usuaris finals. En segon lloc, **AbocadorsBPAPI**, una API[5] REST[6] desenvolupada amb FastAPI [11] que gestiona tota la lògica de negoci, l'autenticació d'usuaris i la persistència de dades a través de PostgreSQL. Finalment, el mòdul d'intel·ligència artificial basat en YOLOv11, entrenat específicament per detectar i classificar vuit tipus de residus: metall, plàstic, vidre, paper i cartró, runa, residus orgànics, mobles i altres.

La solució implementa funcionalitats avançades com un sistema complet de gestió d'usuaris amb registre, autenticació i perfils personalitzats. La funcionalitat principal permet fotografiar abocadors, geolocalitzar-los automàticament mitjançant GPS[7] i enviar denúncies

[4] YOLO: You Only Look Once

[5] API: Application Programming Interface

[6] REST: Representational State Transfer

[7] GPS: Global Positioning System

que inclouen descripció textual i classificació (automàtica, manual o una combinació de les anteriors) de residus. La interfície integra mapes interactius desenvolupats amb *Leaflet* [12] que permeten visualitzar abocadors existents i navegar geogràficament per les denúncies. A més, inclou un sistema d'estadístiques personalitzades que *gamifica*[8] l'experiència de l'usuari, mostrant mètriques de contribució i comparatives regionals.

Un aspecte innovador del projecte és la integració del model d'intel·ligència artificial que analitza automàticament les fotografies i redueix significativament la càrrega manual de categorització, proporcionant dades estructurades valuoses per a anàlisis posteriors. Aquest model anirà millorant amb el temps a mesura que s'obtinguin dades per tal de facilitar cada cop més la tasca d'enregistrament.

Així doncs, es pot categoritzar com una aplicació de processament de dades en temps real amb components d'anàlisi per lots. Mentre que les operacions bàsiques de registre i consulta de denúncies són immediates, el processament d'imatges mitjançant IA pot requerir diversos segons, una latència completament acceptable per al domini d'aplicació on la precisió en la classificació és més important que la velocitat de resposta.

Des del punt de vista formatiu, aquest treball suposa una integració de coneixements adquirits al llarg del grau, incloent-hi l'anàlisi i el disseny de sistemes, el desenvolupament d'aplicacions mòbils, l'aprenentatge automàtic i la gestió de bases de dades. També representa un repte realista amb una aplicació pràctica i potencial de creixement, tant des del punt de vista tècnic com social.

Finalment, cal esmentar que **AbocadorsBP** representa un projecte d'impacte social real el qual en data de publicació d'aquest document està treballant per a unificar-se amb administracions locals del Baix Penedès. L'aplicació no constitueix només un exercici acadèmic, sinó una eina pràctica destinada a ser desplegada en producció per al benefici de la comunitat. Aquest aspecte afegeix una dimensió de responsabilitat social al desenvolupament tècnic, requerint especial atenció a aspectes com la usabilitat, la privacitat de dades i la robustesa del sistema.

Tot el codi relacionat amb el projecte es mantindrà en repositoris privats per motius de seguretat i amb vista a una posterior explotació del projecte. Tanmateix, el tribunal pot sol·licitar-ne l'accés prèvia petició formal a l'estudiant. Si el tribunal vol obtenir accés a l'aplicació, la poden trobar a la App Store sota el nom d'"*abocadorsbp*" en cas de tenir iOS. En cas d'utilitzar Android, poden posar-se en contacte amb l'alumne i proporcionar el correu utilitzat a la Google Play Store. L'alumne respondrà amb un enllaç cap a la Play Store des d'on el tribunal podrà descarregar l'aplicació.

[8] Gamificar: aplicar elements i mecàniques pròpies dels jocs (com punts, nivells o recompenses) en contextos no lúdics per augmentar la motivació, l'adhesió i la interacció dels usuaris.

3 Requisits

Els requisits del projecte es divideixen en funcionals i no funcionals. Els requisits funcionals ens permeten conèixer quines funcionalitats - és a dir, casos d'ús - podem trobar com a usuaris o administradors a l'hora de fer servir l'aplicació, mentre que els requisits no funcionals defineixen restriccions, característiques de qualitat i rendiment per a l'aplicació.

3.1 Requisits Funcionals

L'aplicació disposa de perfils bàsics d'usuari però també existeixen els perfils d'administrador, que estenen els d'usuari. És per això que a l'especificació textual, els casos d'ús específics per als administradors s'indiquen amb un asterisc vermell (*).

3.1.1 Especificació Textual dels Casos d'Ús

RF-01. Iniciar Sessió

- Resum: L'usuari introdueix les seves credencials per accedir a l'aplicació.
- Paràmetres d'entrada: Email, contrasenya.
- Paràmetres de sortida: Dades d'usuari, confirmació d'autenticació.
- Actors: Usuari, sistema.
- Precondició: L'usuari té un compte registrat.
- Postcondició: L'usuari està autenticat i les seves dades s'emmagatzemen localment.
- Procés normal principal:
  1. L'usuari introdueix el seu email i contrasenya.
  2. El sistema valida les credencials contra la base de dades.
  3. El sistema retorna les dades de l'usuari (email, nom, comarca, etc.).
  4. El sistema emmagatzema les dades de l'usuari localment.
  5. L'usuari és redirigit a la pantalla anterior (Mapa, Càmera o Perfil).

  Alternatives de procés i excepcions:

  3a. L'usuari no existeix o la contrasenya és incorrecta.

  3a1. El sistema mostra un missatge d'error.

  3a2. El sistema torna al pas 1.

RF-02. Registrar usuari

- Resum: Crea un nou compte d'usuari al sistema.
- Paràmetres d'entrada: Email, contrasenya.
- Paràmetres de sortida: Confirmació de registre.
- Actors: Usuari, sistema.
- Precondició: L'email no està registrat prèviament.
- Postcondició: Es crea un nou compte d'usuari al sistema.
- Procés normal principal:
  1. L'usuari introdueix email i contrasenya (dues vegades).
  2. El sistema valida que l'email no existeix.
  3. El sistema genera un *token* de verificació.

4. El sistema envia un email mitjançant SMTP[9] [13] amb l'enllaç de verificació.
5. L'usuari rep confirmació d'enviament d'email.
6. L'usuari accedeix a l'enllaç de verificació.
7. El sistema crea el nou compte.
8. Es mostra confirmació de registre.
9. L'usuari és redirigit a la pantalla d'inici de sessió.

Alternatives de procés i excepcions:

1a. La contrasenya no coincideix.

1a1. El sistema mostra un missatge d'error.

1a2. El sistema torna al pas 1.

2a. Ja existeix un compte registrat sota l'email proporcionat.

2a1. El sistema mostra un missatge d'error.

2a2. El sistema torna al pas 1.

3a. L'usuari no accedeix a l'enllaç o ho fa passat el termini d'una (1) hora.

3a1. El sistema considera invàlid el *token* de verificació.

3a2. El sistema no registra cap usuari al sistema.

RF-03. Recuperar contrasenya

- Resum: L'usuari sol·licita restablir la seva contrasenya.
- Paràmetres d'entrada: Email.
- Paràmetres de sortida: Enllaç de recuperació enviat per email.
- Actors: Usuari, sistema.
- Precondició: L'usuari té un compte registrat.
- Postcondició: S'envia un email amb enllaç per restablir la contrasenya.
- Procés normal principal:
  1. L'usuari introdueix el seu email.
  2. El sistema verifica que l'email existeix.
  3. El sistema genera un *token* de recuperació.
  4. El sistema envia un email mitjançant SMTP amb l'enllaç de recuperació.
  5. L'usuari rep confirmació que s'ha enviat l'email.
  6. L'usuari accedeix a l'enllaç de recuperació i introdueix la contrasenya nova (dues vegades).
  7. El sistema persisteix el hash de la nova contrasenya a la base de dades.
  8. El sistema informa a l'usuari de que ja pot iniciar sessió.

  Alternatives de procés i excepcions:

  2a. No existeix cap compte associat a l'email rebut.

  2a1. El sistema mostra un missatge d'error.

[9] SMTP: Simple Mail Transfer Protocol

2a2. El sistema torna al pas 1.

6a. L'usuari no accedeix a l'enllaç o ho fa passat el termini d'una (1) hora.

6a1. El sistema considera invàlid el *token* de recuperació.

6a2. El sistema no fa cap canvi.

RF-04. Editar perfil

- Resum: L'usuari modifica la seva informació personal o foto de perfil.
- Paràmetres d'entrada: Nom d'usuari, comarca, municipi, foto de perfil, email.
- Paràmetres de sortida: Confirmació d'actualització.
- Actors: Usuari, sistema
- Precondició: L'usuari està autenticat.
- Postcondició: La informació del perfil s'actualitza al sistema.
- Procés normal principal:
  1. L'usuari accedeix a la pantalla d'edició de perfil.
  2. El sistema carrega les dades actuals del perfil.
  3. L'usuari modifica els camps desitjats (inclosa la foto de perfil).
  4. L'usuari accepta els canvis fets.
  5. El sistema actualitza la informació al servidor.
  6. Es mostra confirmació d'actualització.
  7. L'usuari és redirigit a la vista del perfil.

Alternatives de procés i excepcions:

3a. L'usuari prem el botó per a canviar la foto de perfil.

3a1. El sistema pregunta a l'usuari si vol fer foto o buscar a la galeria.

3a2. El sistema verifica si existeixen permisos per a accedir a la càmera o a la galeria o es demana a l'usuari que activi els permisos.

3a3. L'usuari escull o fa una foto.

3a4. L'usuari retalla la foto per a que sigui quadrada.

3a5. L'usuari accepta el canvi de foto.

3a6. El sistema segueix al pas 3.

3b. L'usuari desplega la llista de comarques.

3b1. El sistema mostra la llista de comarques.

3b2. L'usuari escull la seva comarca.

3b3. El sistema segueix al pas 3.

3c. L'usuari desplega la llista de municipis.

3c1. El sistema verifica si l'usuari té una comarca definida.

3c2. El sistema mostra la llista de municipis de la comarca definida o la llista de tots els municipis.

4a. L'usuari no accepta els canvis fets i simplement torna enrere.

4a1. El sistema no actualitza informació d'usuari.

4a2. L'usuari és redirigit a la pantalla del perfil.

RF-05. Tancar Sessió

- Resum: L'usuari finalitza la seva sessió a l'aplicació.
- Paràmetres d'entrada: Cap.
- Paràmetres de sortida: Confirmació de tancament de sessió.
- Actors: Usuari, sistema
- Precondició: L'usuari està autenticat
- Postcondició: La sessió d'usuari acaba i es netegen les dades locals.
- Procés normal principal:
  1. L'usuari selecciona l'opció de tancar sessió
  2. El sistema elimina totes les dades emmagatzemades localment
  3. L'usuari és redirigit a la pantalla d'inici de sessió

RF-06. Consultar perfil d'usuari

- Resum: Visualitzar informació i estadístiques d'un usuari (propi o d'altre)
- Paràmetres d'entrada: Email de l'usuari a consultar
- Paràmetres de sortida: Dades del perfil, estadístiques de denúncies.
- Actors: Usuari, sistema.
- Precondició: En cas de consultar el perfil propi, l'usuari ha d'estar autenticat. En cas de consultar un altre perfil, cap.
- Postcondició: Es mostra la informació del perfil sol·licitat.
- Procés normal principal:
  1. L'usuari accedeix al perfil (propi des de la pestanya o d'altre des d'una denúncia).
  2. El sistema carrega les dades del perfil (imatge, nom, comarca, municipi).
  3. El sistema obté les estadístiques de denúncies de l'usuari. (denúncies totals, a la seva comarca i al seu municipi).
  4. Es mostra la informació del perfil amb gràfics i estadístiques.
  5. L'usuari pot navegar a veure les denúncies de l'usuari (RF-07)

  Alternatives de procés i excepcions:

  2a. L'usuari seleccionat manca dades (usuari, comarca, municipi, foto)

  2a1. El sistema mostra un *placeholder*[10].

  2a2. El sistema segueix al pas 3.

RF-07. Consultar denúncies d'usuari

- Resum: Visualitzar la llista paginada de denúncies d'un usuari específic.
- Paràmetres d'entrada: Email de l'usuari, número de pàgina.
- Paràmetres de sortida: Llista de denúncies amb imatges i detalls.
- Actors: Usuari, sistema.

[10] Placeholder: Element temporal o text indicatiu que ocupa l'espai d'una dada o component pendent de ser proporcionat o definit per l'usuari o pel sistema.

- Precondició: En cas de consultar les denúncies pròpies, l'usuari ha d'estar autenticat. En cas de consultar les d'un altre perfil, cap.
- Postcondició: Es mostra la llista de denúncies de l'usuari especificat.
- Procés normal principal:
  1. L'usuari accedeix a la secció de denúncies
  2. El sistema carrega les denúncies de la pàgina actual (inicialment la 1)
  3. El sistema carrega les imatges de forma asíncrona mentre que mostra les dades dels abocadors.
  4. L'usuari pot navegar entre pàgines i especificar la pàgina exacte.
  5. L'usuari pot consultar detalls d'una denúncia específica tocant-la
  6. Es mostra un mapa amb la ubicació de la denúncia seleccionada i més informació rellevant.

RF-08*. Gestionar abocadors pendents

- Resum: Aprovar o rebutjar abocadors pendents registrats per usuaris.
- Paràmetres d'entrada: Descripció de l'abocador, llista de residus.
- Paràmetres de sortida: Avís en cas de rebutjar, cap en cas d'acceptar.
- Actors: Administrador, sistema.
- Precondició: L'usuari ha d'estar autenticat i ha de ser un administrador.
- Postcondició: L'estat de l'abocador s'actualitza (acceptat/rebutjat).
- Procés normal principal:
  1. L'administrador accedeix a la vista de gestió d'abocadors pendents.
  2. El sistema carrega els abocadors pendents de forma paginada.
  3. L'administrador revisa la imatge i detalls de l'abocador.
  4. L'administrador edita la descripció si cal.
  5. L'administrador afegeix o modifica els residus detectats si cal.
  6. L'administrador decideix acceptar o rebutjar l'abocador.
  7. El sistema actualitza l'estat de l'abocador.
  8. L'administrador pot continuar amb el següent abocador pendent

RF-09. Consultar mapa d'abocadors

- Resum: Visualitzar tots els abocadors denunciats en un mapa interactiu.
- Paràmetres d'entrada: Comarca, municipi, estat dels abocadors, tipus de mapa, coordenades de la capsa de vista del mapa.
- Paràmetres de sortida: Mapa amb marcadors (agrupats, individuals o ambdós) d'abocadors dins la vista del mapa, estadístiques de l'àrea
- Actors: Usuari, sistema.
- Precondició: Cap.
- Postcondició: Es mostra el mapa amb els abocadors filtrats.
- Procés normal principal:
  1. L'usuari accedeix a la pestanya del mapa.
  2. El sistema obté la ubicació actual de l'usuari.
  3. El sistema obté la capsa englobant de la zona visible del mapa
  4. El sistema carrega els abocadors de la zona visible.
  5. Es mostren estadístiques de l'àrea seleccionada.
  6. L'usuari pot centrar el mapa en la seva ubicació actual.

Alternatives de procés i excepcions:

6a. L'usuari aplica filtres per comarca, municipi o estat.

6a1. El sistema actualitza els marcadors segons els filtres.

6a2. Repeteix el pas 6.

6b. L'usuari canvia el tipus de mapa (estàndard, clar, fosc, satèl·lit).

6b1. El sistema carrega el fons del mapa.

6b2. Repeteix el pas 6.

RF-10. Consultar informació d'abocador al mapa

- Resum: Visualitzar detalls complets d'un abocador específic des del mapa.
- Paràmetres d'entrada: ID de l'abocador seleccionat
- Paràmetres de sortida: Detalls complets de l'abocador, nom del denunciant.
- Actors: Usuari, sistema
- Precondició: L'usuari està al mapa i hi ha abocadors visibles.
- Postcondició: Es mostra la informació detallada de l'abocador seleccionat.
- Procés normal principal:
  1. L'usuari toca un marcador d'abocador al mapa.
  2. El sistema carrega els detalls de l'abocador.
  3. El sistema obté la informació del denunciant.
  4. El sistema carrega els residus associats a l'abocador.
  5. Es mostra un modal amb tota la informació detallada.
  6. El sistema mostra la informació de l'abocador seleccionat.
  7. L'usuari pot tancar el modal per tornar al mapa (RF-09).

  Alternatives de procés i excepcions:

  7a. L'usuari accedeix al perfil del denunciant (RF-06).

RF-11. Enregistrar nova denúncia d'abocador

- Resum: Crear una nova denúncia d'abocador il·legal amb detecció automàtica de residus i ubicació.
- Paràmetres d'entrada: Foto d'abocador, ubicació, residus, descripció opcional.
- Paràmetres de sortida: Confirmació de denúncia creada.
- Actors: Usuari, sistema.
- Precondició: L'usuari està autenticat i ha donat permisos de càmera i ubicació
- Postcondició: Es crea una nova denúncia d'abocador al sistema.
- Procés normal principal:
  1. L'usuari accedeix a la pestanya de càmera.
  2. L'usuari fa una foto de l'abocador o l'obté de la galeria.
  3. El sistema detecta automàticament els tipus de residus mitjançant IA.
  4. El sistema obté automàticament la ubicació actual de l'usuari.
  5. El sistema detecta automàticament comarca i municipi.
  6. L'usuari pot afegir una descripció opcional
  7. L'usuari confirma i envia la denúncia
  8. El sistema crea la denúncia amb estat "pendent"

9. Es mostra confirmació de denúncia enviada

Alternatives de procés i excepcions:

3a. L'usuari clica el botó "+" per a afegir o eliminar residus a la denúncia.

3a1. Es mostra un modal amb la llista de residus.

3a2. L'usuari selecciona o des selecciona els residus que considera.

3a3. L'usuari tanca el modal amb la llista de residus.

4a. L'usuari clica el botó per a modificar la ubicació manualment.

4a1. Es mostra un modal amb un mapa i dos camps d'entrada de text.

4a2. L'usuari pot desplaçar-se per el mapa i clicar la ubicació al mapa on vol registrar l'abocador.

4a3. L'usuari pot introduir les coordenades (latitud i longitud) manualment.

RF-12*. Editar abocador

- Resum: Editar la informació d'un abocador ja existent.
- Paràmetres d'entrada (tots opcionals): coordenades, comarca, municipi, residus, descripció.
- Paràmetres de sortida: Confirmació de canvis guardats.
- Actors: Administrador, sistema.
- Precondició: L'usuari està autenticat i és administrador.
- Postcondició: Es guarden els canvis fets sobre l'abocador al sistema.
- Procés normal principal:
  1. L'administrador toca un marcador d'abocador al mapa.
  2. L'administrador toca el botó per editar l'abocador (botó del llapis).
  3. El sistema activa el mode d'edició i mostra botons abans ocults.
  4. L'administrador pot modificar localització (coordenades, comarca i municipi), descripció i residus detectats.
  5. L'administrador confirma els canvis fets.
  6. El sistema persisteix els canvis fets sobre aquell abocador.
  7. Es mostra missatge de confirmació dels canvis.

Alternatives de procés i excepcions:

4a. L'administrador clica el botó "+" per a afegir o eliminar residus a la denúncia.

4a1. Es mostra un modal amb la llista de residus.

4a2. L'usuari selecciona o des selecciona els residus que considera.

4a3. L'usuari tanca el modal amb la llista de residus.

4b. L'administrador clica el botó per a modificar la ubicació manualment.

4b1. Es mostra un modal amb un mapa i dos camps d'entrada de text.

4b2. L'usuari pot desplaçar-se per el mapa i clicar la ubicació on vol registrar l'abocador.

4b3. L'usuari pot introduir coordenades manualment.

4c. L'administrador clica el camp de la descripció.

4c1. L'administrador modifica la descripció al seu gust.

5a. L'administrador cancel·la els canvis fets.

5a1. Es mostra missatge de confirmació de cancel·lació

5a2. Acaba el cas d'ús.

RF-13. Retirar Abocador

- Resum: Marcar un abocador com a retirat o notificar-ho a un administrador.
- Paràmetres d'entrada: nom d'abocador, ID d'abocador, comarca, municipi.
- Paràmetres de sortida: Confirmació d'acció acabada.
- Actors: Usuari, sistema.
- Precondició: Cap.
- Postcondició: Es guarden els canvis fets sobre l'abocador al sistema o s'envia un correu a un administrador.
- Procés normal principal:
  1. L'usuari toca un marcador d'abocador al mapa.
  2. L'usuari toca el botó per marcar l'abocador com a retirat (botó de la fulla).
  3. El sistema pregunta a l'usuari si realment vol notificar aquell abocador com a retirat.
  4. L'usuari accepta que vol notificar el canvi.
  5. El sistema envia un correu automatitzat al correu d'administrador abocadorsbpdes@gmail.com amb la informació de l'abocador.
  6. Es mostra un missatge de confirmació d'enviament de notificació.

  Alternatives de procés i excepcions:

  3a. El sistema detecta que l'usuari és un Administrador i l'hi pregunta si realment vol aplicar aquest canvi d'estat sobre l'abocador.

  3a1. L'administrador accepta

  3a2. El sistema persisteix els canvis a la base de dades.

  4a. L'usuari no accepta que vol notificar el canvi.

  4a1. El sistema no fa res i el cas d'ús acaba.

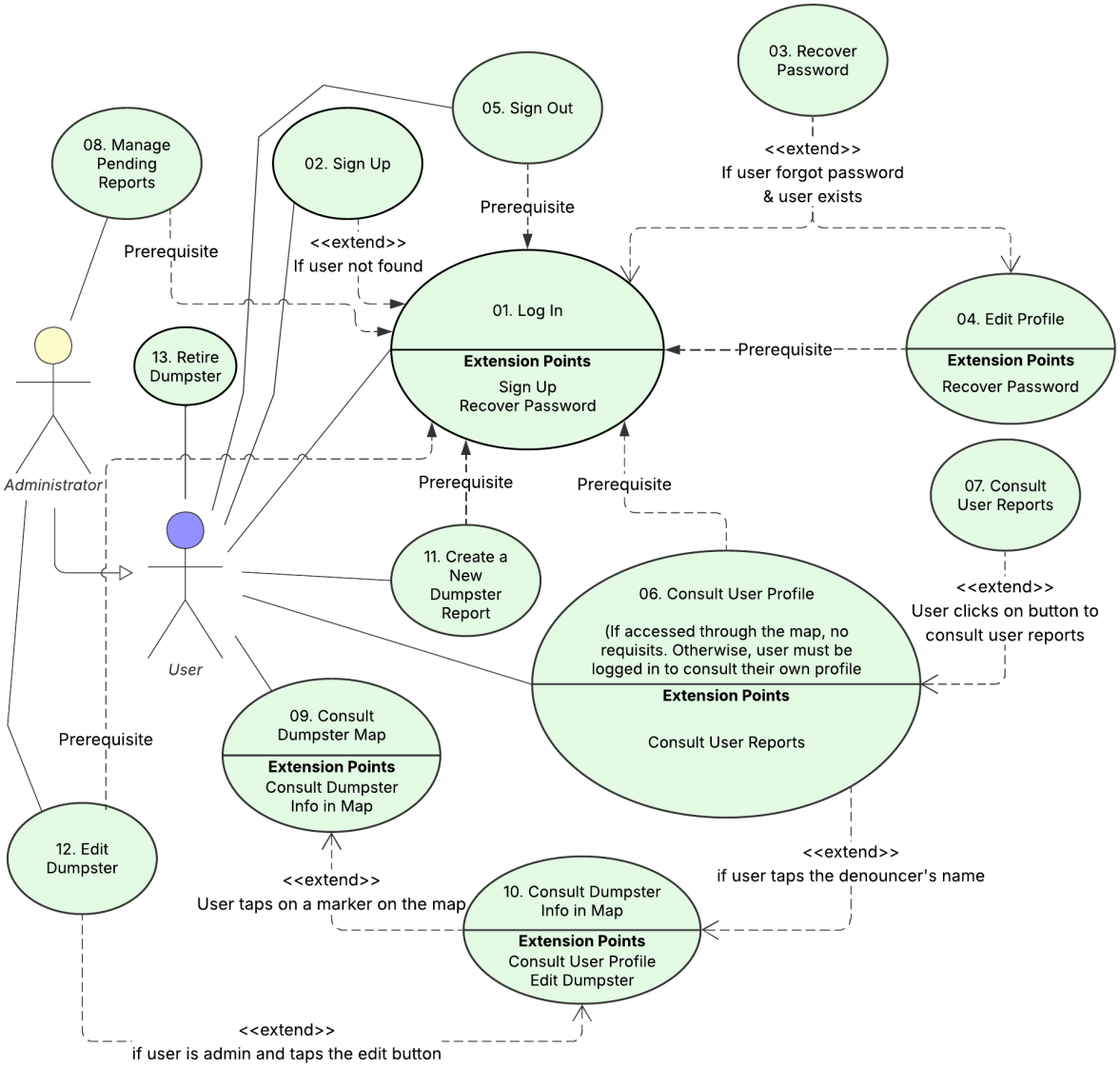
3.1.2 Diagrama de Casos d'Ús

Per tal de poder visualitzar les funcionalitats de l'aplicació, s'ha elaborat un diagrama de casos d'ús mitjançant l'eina LucidChart.

Inicialment veiem que els casos d'ús que necessiten que l'usuari estigui autenticat al sistema són els següents:

- RF-04. Editar Perfil (*04. Edit Profile*).
- RF-05. Tancar Sessió (*05. Sign Out*).
- RF-06. Consultar perfil d'usuari (*06. Consult User Profile*) però només en cas de que s'intenti consultar el perfil propi, no des del mapa.
- RF-08. Gestionar abocadors pendents (*08. Manage Pending Reports*).
- RF-11. Enregistrar nova denúncia d'abocador (*11. Create New Dumpster Report*).
- RF-12. Editar abocador (*12. Edit Dumpster*)

A més els casos d´ús RF-08. Gestionar abocadors pendents (*08. Manage Pending Reports*) i RF-012. Editar abocador (*12. Edit Dumpster*) són els únics que no poden ser usats per un usuari no administrador.



Il·lustració 1. Diagrama de casos d'ús

3.2 Requisits No Funcionals

Els requisits no funcionals permeten especificar les característiques del funcionament de l'aplicació, és a dir, com ha de comportar-se el sistema. Aquest apartat doncs defineix les restriccions, característiques de qualitat i rendiment que ha de tenir aquest.

REQUERIMENT	DESCRIPCIÓ
Usabilitat	L'aplicació ha de tenir una navegació clara amb pantalles simples, indicadors de càrrega, missatges d'èxit i error, etc.
Rendiment	Les pantalles han de carregar-se en el menor temps possible i fer servir *lazy loading*[11] i memòria cau d'imatges per evitar càrregues lentes. A més, les imatges han de comprimir-se per ocupar el menor espai possible.
Seguretat	L'ús d'*AsyncStorage*[12] permetrà tenir emmagatzematge segur local i es verificaran totes les credencials abans d'accedir a funcionalitats. Per tant, el sistema mantindrà de manera privada les credencials i informació sensible dels usuaris.
Fiabilitat i Disponibilitat	L'aplicació ha d'estar operativa al llarg de l'any i en qualsevol moment del dia mentre manté la integritat de les dades.
Escalabilitat	L'aplicació ha de mantenir bon rendiment independentment del nombre d'usuaris i els canvis d'infraestructura.
Mantenibilitat	El disseny estarà pensat per a facilitar el manteniment de l'aplicació i la documentació servirà de guia per a aquesta.
Privacitat	L'aplicació vetllarà per la confidencialitat de les dades dels usuaris i s'encarregarà d'encriptar totes les dades que passin per la xarxa mitjançant HTTPS[13].
Precisió	El sistema ha de ser capaç de detectar amb precisió la ubicació de l'usuari i la comarca i municipi en els que es troba. A més, el model de reconeixement de residus ha d'ajudar més que molestar a l'usuari per a considerar-se útil.
Robustesa	El sistema depurarà qualsevol entrada d'usuari per a assegurar la seguretat de les dades i del sistema mateix.
Compatibilitat	L'aplicació ha de ser compatible a curt termini tant per a Android com per a iOS i a llarg termini per a Web.

Taula 1. Requeriments no funcionals

[11] Lazy Loading: Tècnica d'optimització que difereix la càrrega de recursos fins al moment en què són necessaris, amb l'objectiu de reduir el temps inicial de càrrega i millorar el rendiment de l'aplicació.

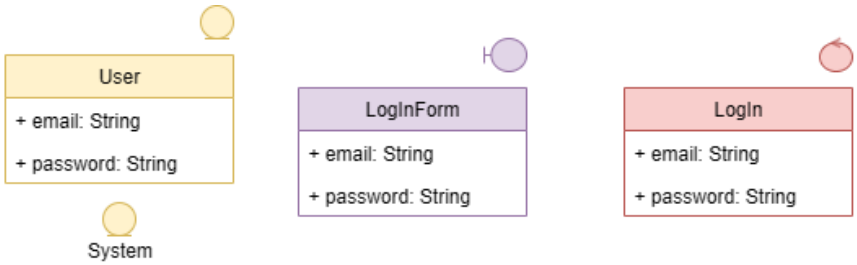
[12] AsyncStorage: Llibreria d'Expo que proporciona emmagatzematge asíncron, no encriptat i persistent de parelles clau-valor.

[13] HTTPS: HyperText Transfer Protocol Secure

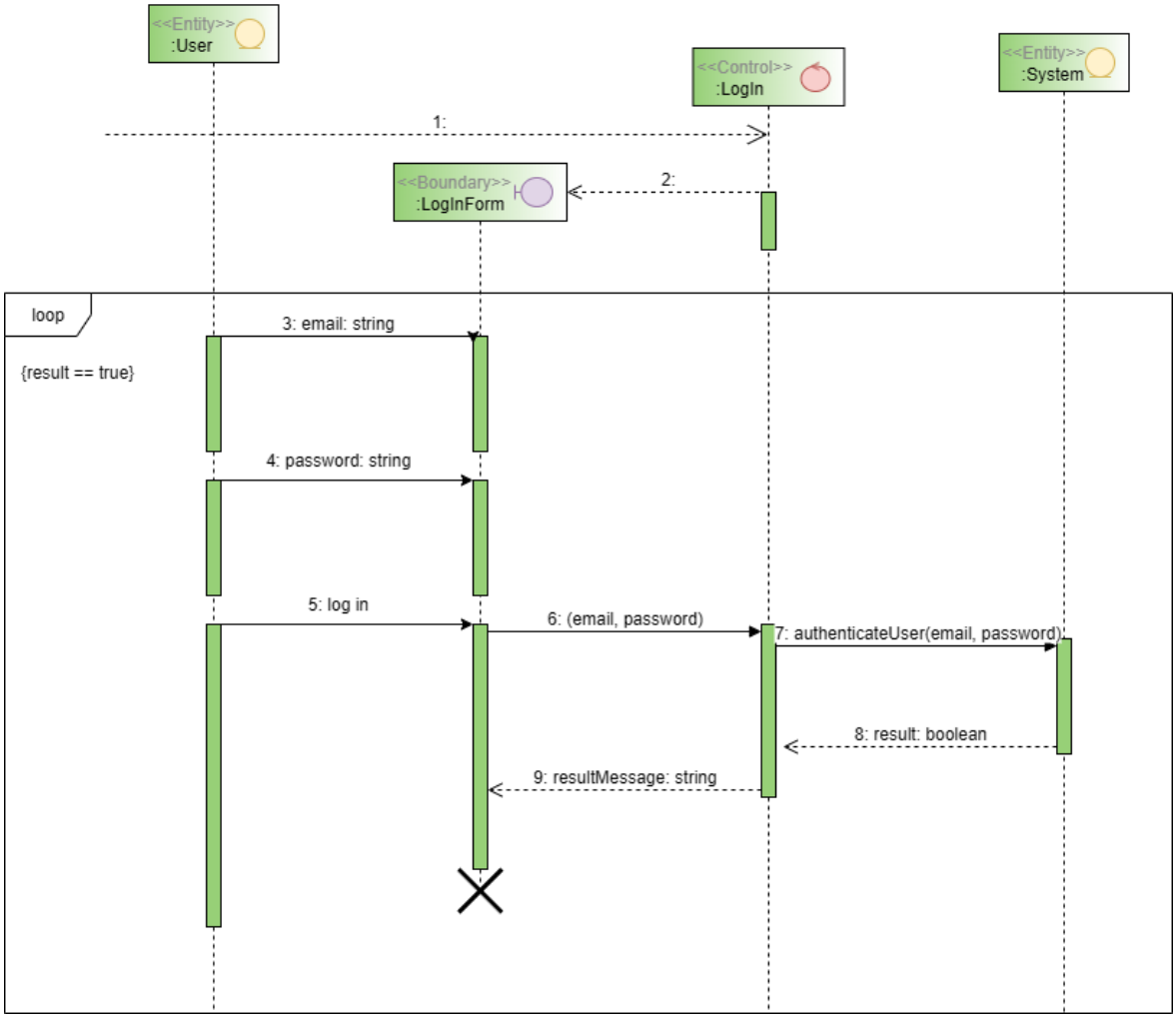
4 Anàlisi dels Requisits Funcionals

Els següents diagrames UML[14] de classes i de seqüència ens permeten analitzar els requisits funcionals esmentats anteriorment d'una manera més visual. Per a generar aquests diagrames s'ha fet servir l'eina Draw.io però donant un estil visual semblant a l'eina MagicDraw per a millorar-ne la llegibilitat.

RF-01. Iniciar Sessió



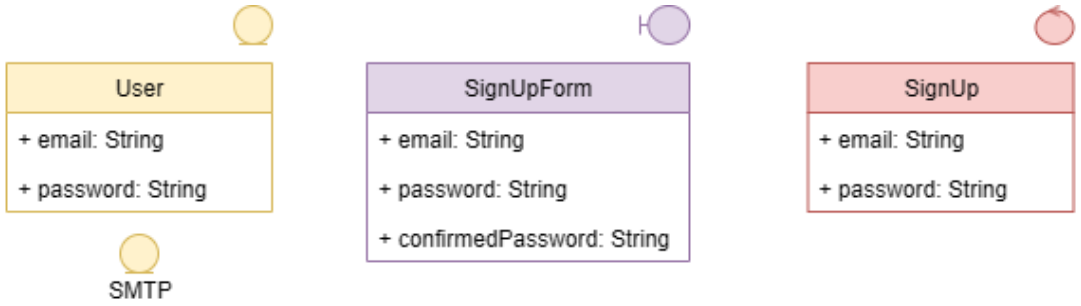
Il·lustració 2. Diagrama de classes del cas d'ús 1



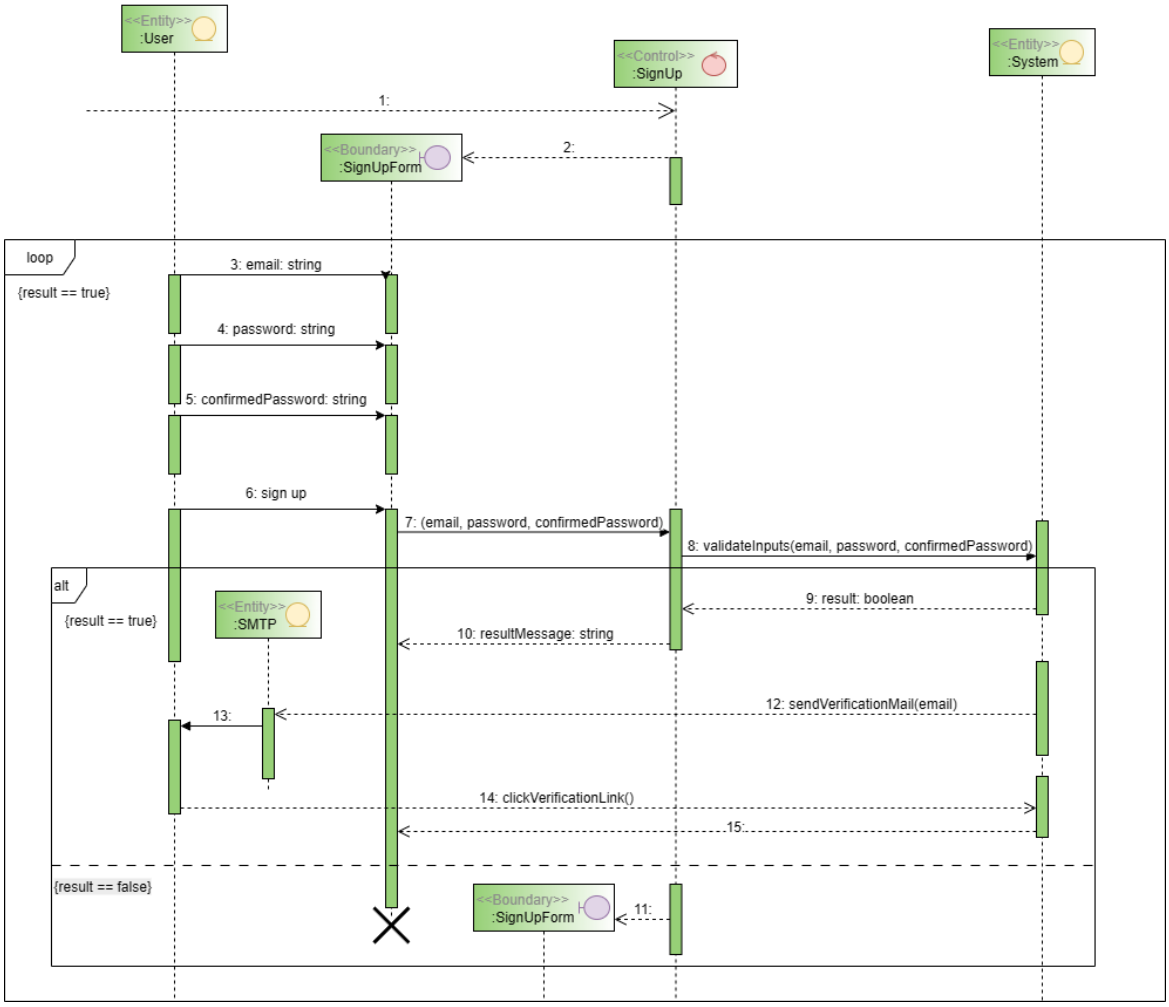
Il·lustració 3. Diagrama de seqüència del cas d´ús 1

[14] UML: Unified Modeling Language

RF-02. Registrar usuari

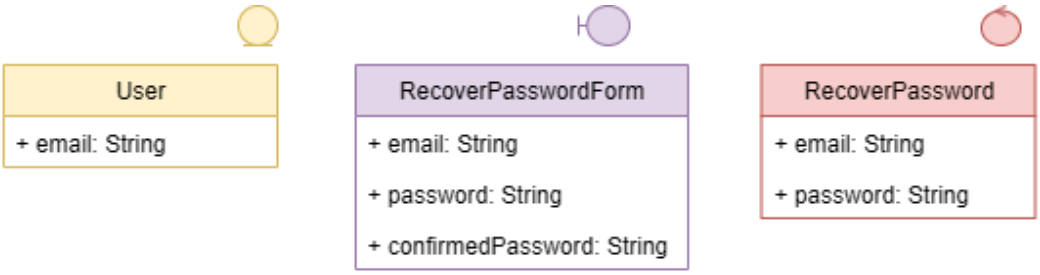


Il·lustració 4. Diagrama de classes del cas d'ús 2

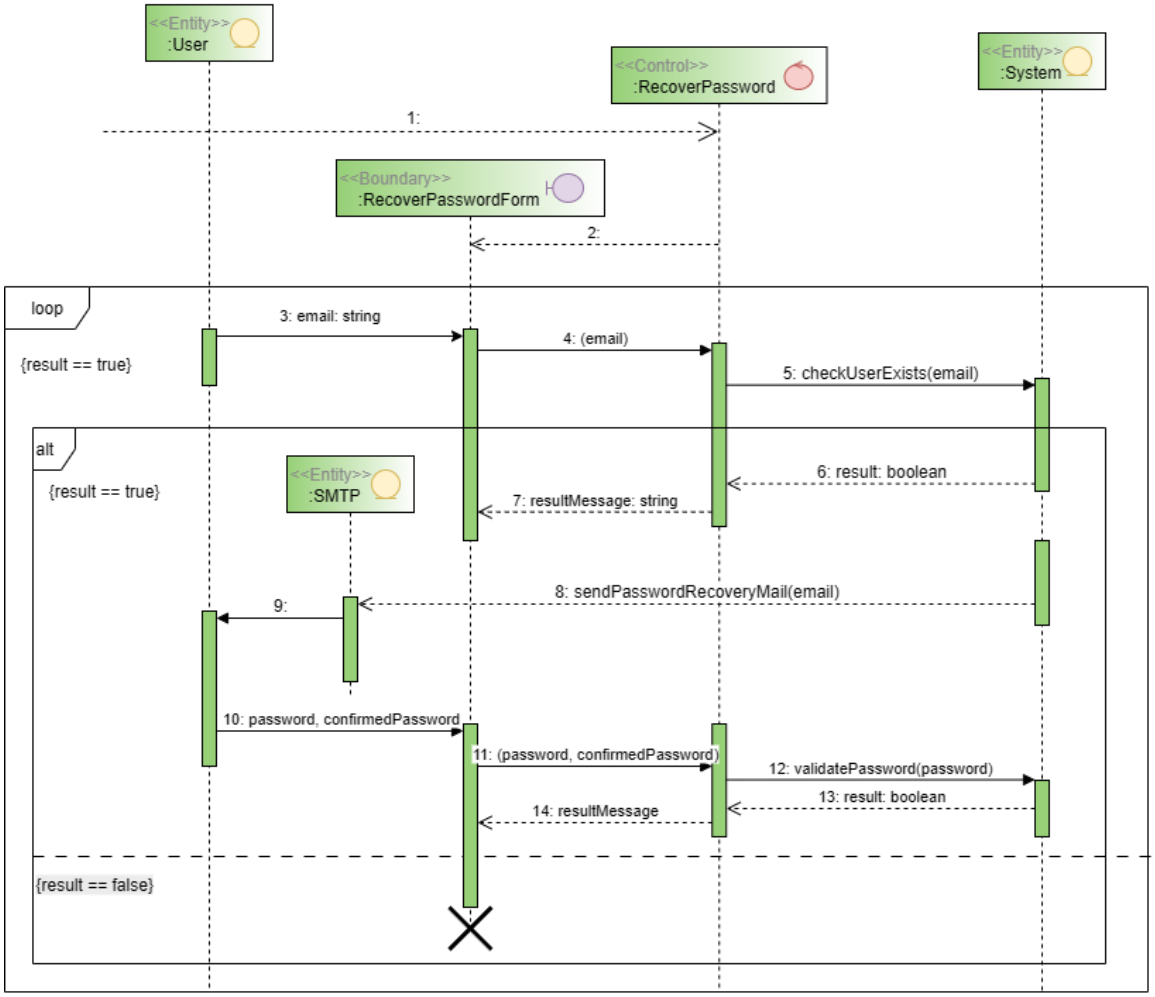


Il·lustració 5. Diagrama de seqüència del cas d´ús 2

RF-03. Recuperar contrasenya

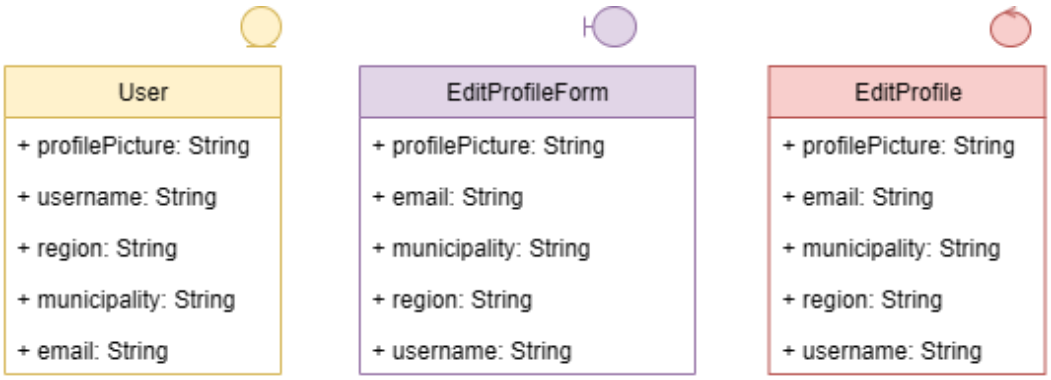


Il·lustració 6. Diagrama de classes del cas d'ús 3

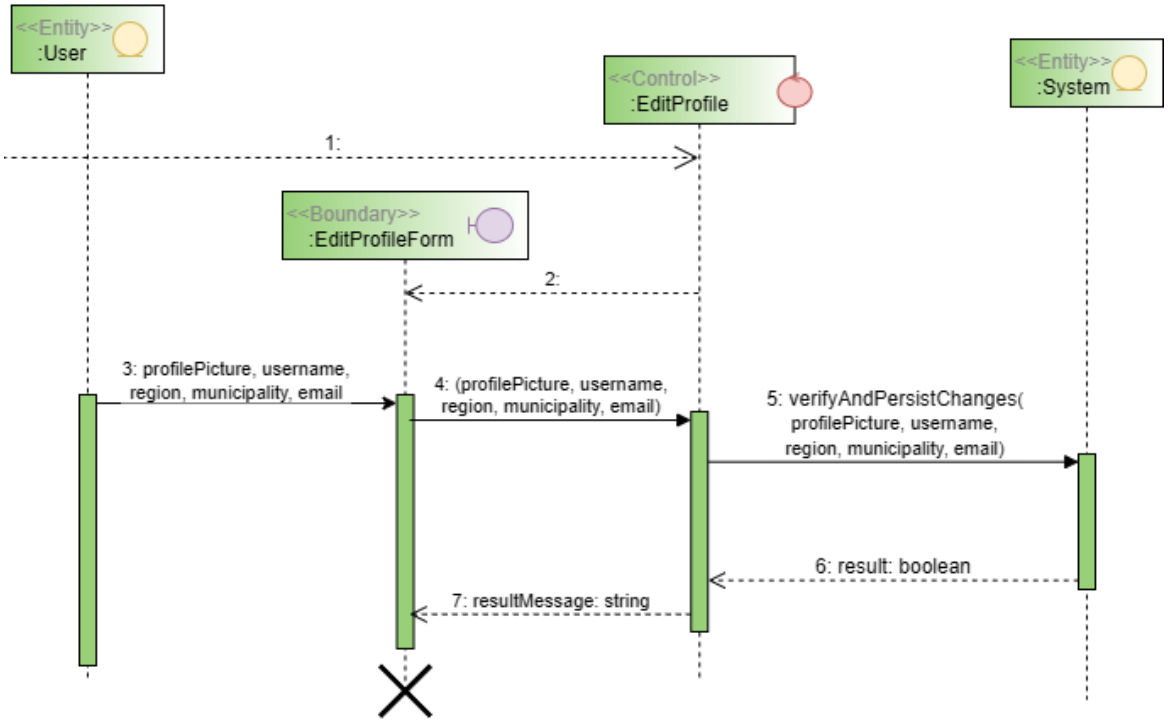


Il·lustració 7. Diagrama de seqüència del cas d´ús 3

RF-04. Editar perfil

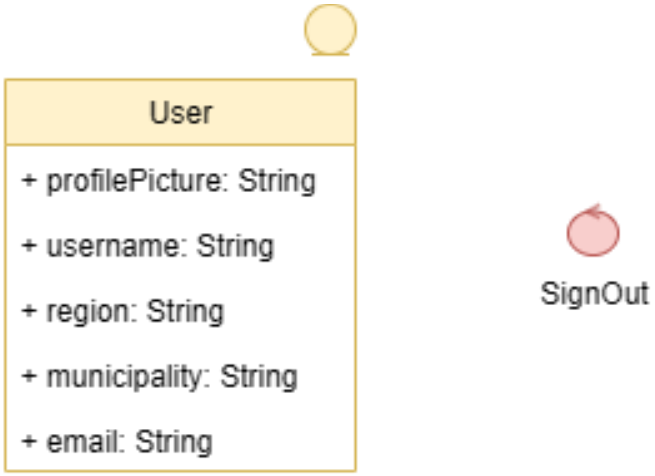


Il·lustració 8. Diagrama de classes del cas d'ús 4

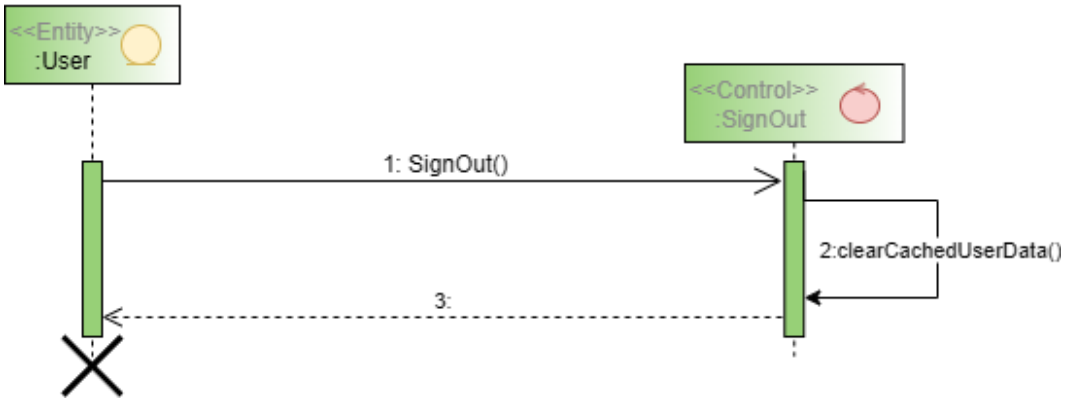


Il·lustració 9. Diagrama de seqüència del cas d´ús 4

RF-05. Tancar Sessió

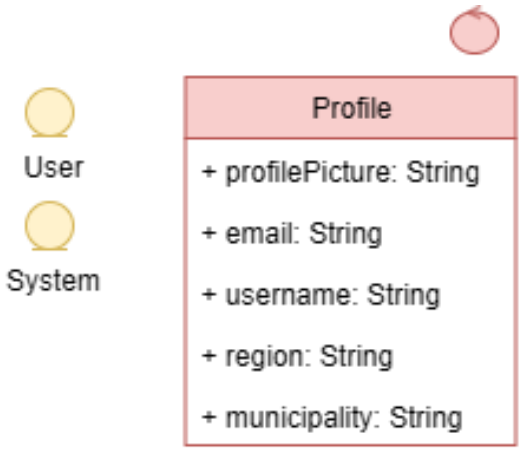


Il·lustració 10. Diagrama de classes del cas d'ús 5

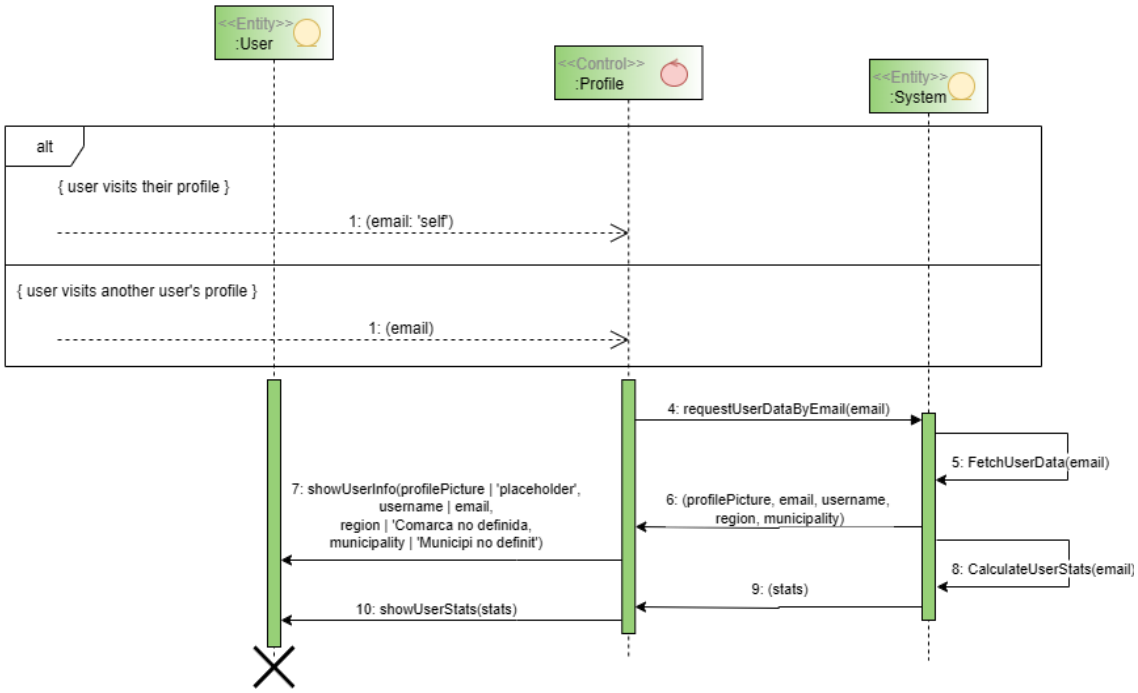


Il·lustració 11. Diagrama de seqüència del cas d´ús 5

RF-06. Consultar perfil d'usuari

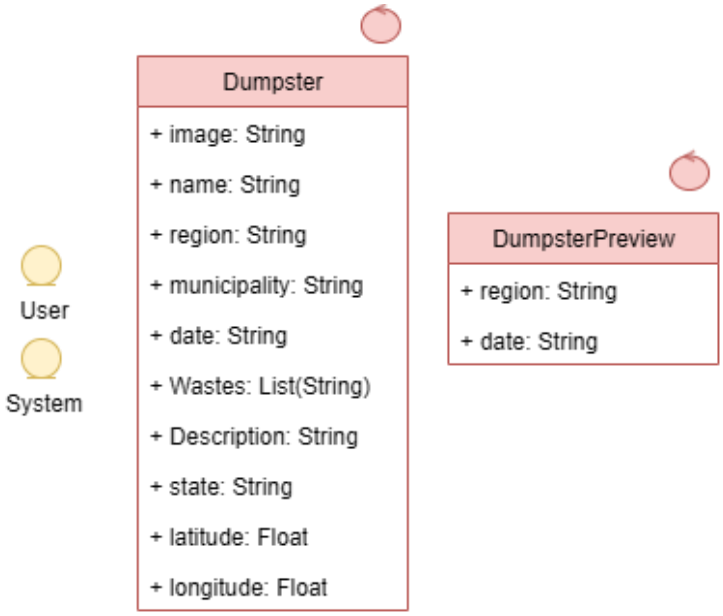


Il·lustració 12. Diagrama de classes del cas d'ús 6

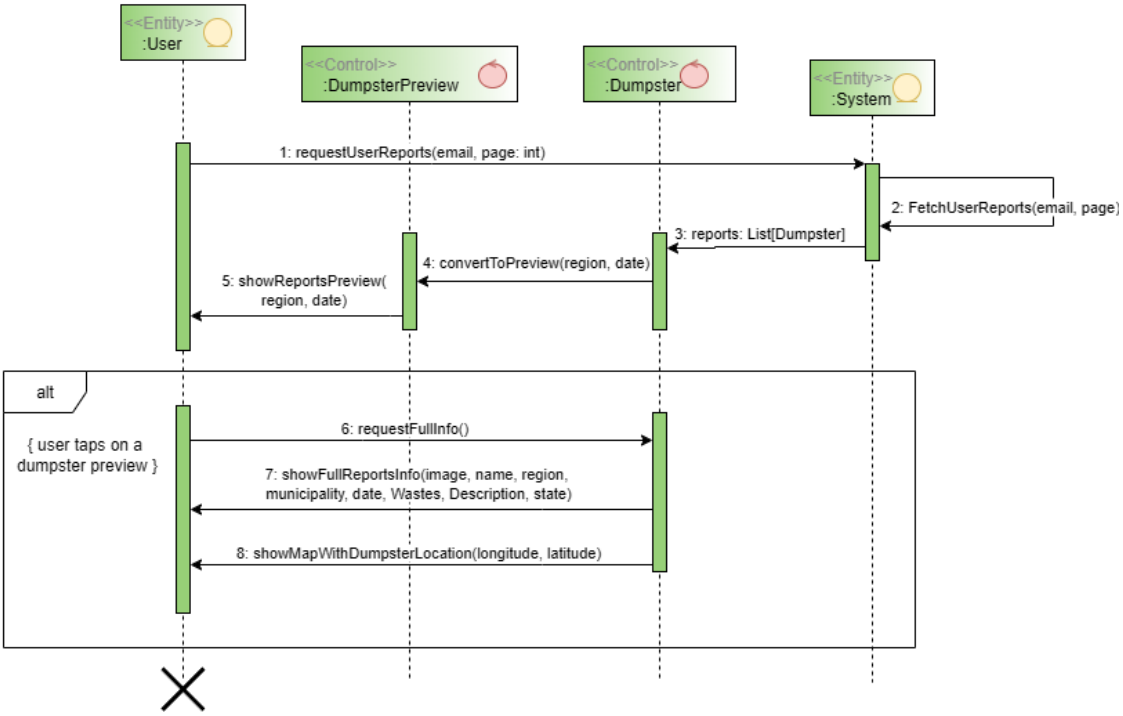


Il·lustració 13. Diagrama de seqüència del cas d´ús 6

RF-07. Consultar denúncies d'usuari

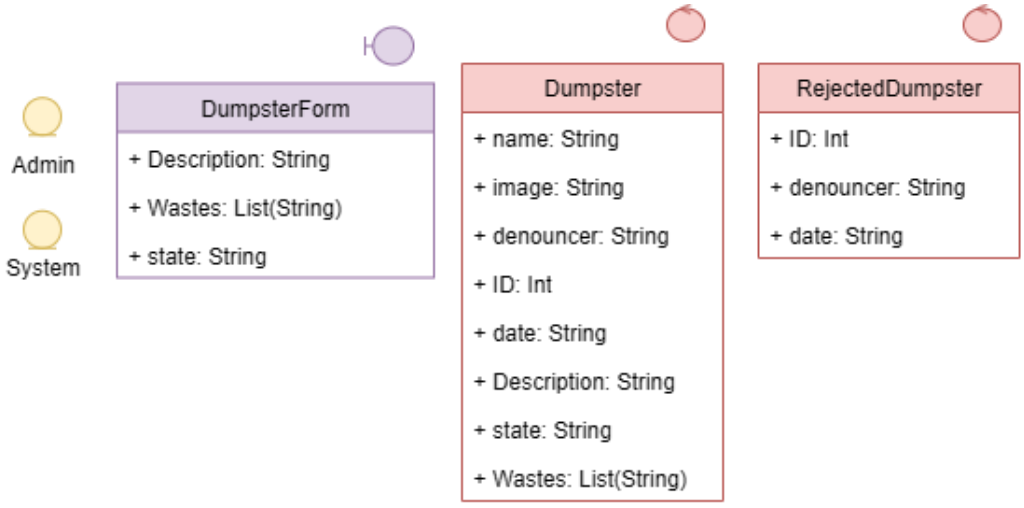


Il·lustració 14. Diagrama de classes del cas d'ús 7

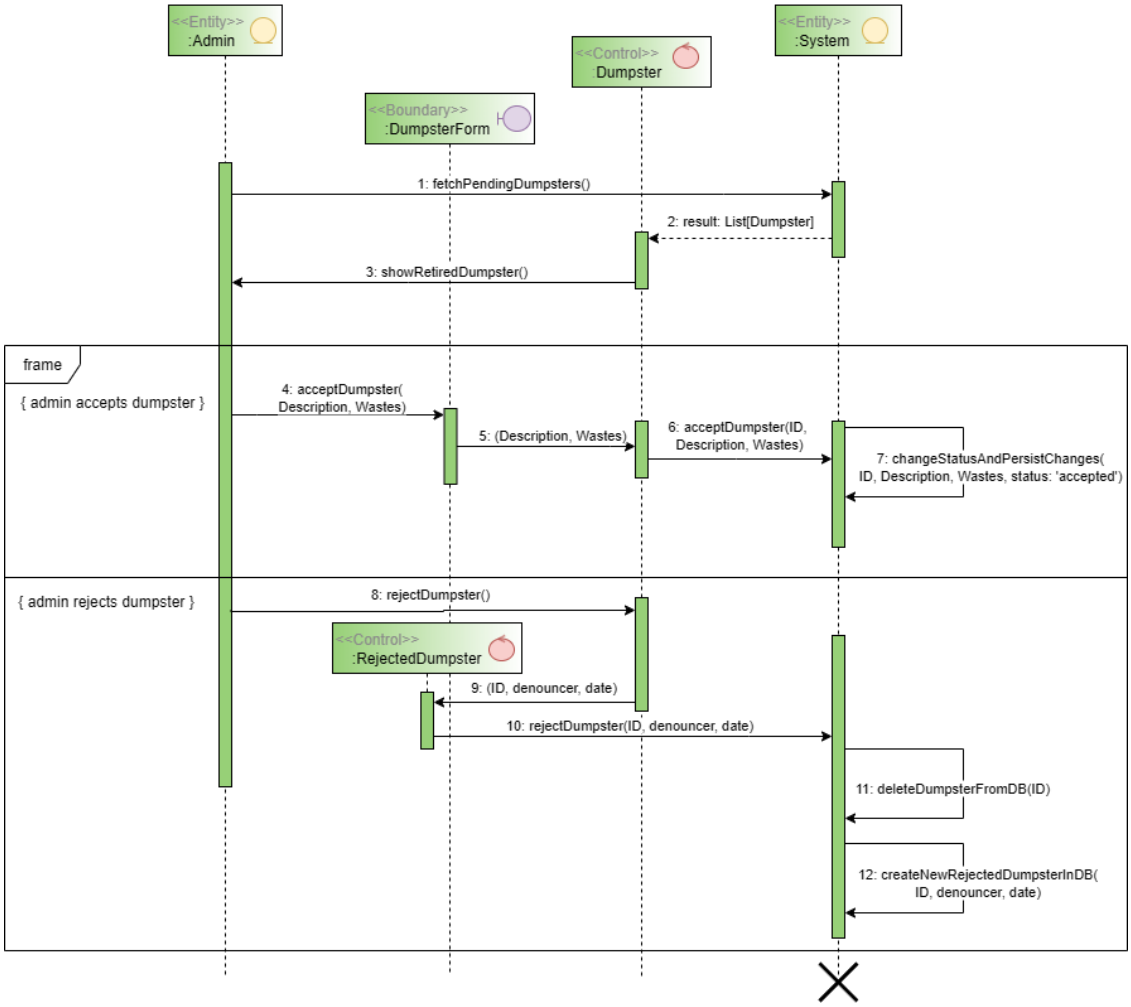


Il·lustració 15. Diagrama de seqüència del cas d´ús 7

RF-08*. Gestionar abocadors pendents

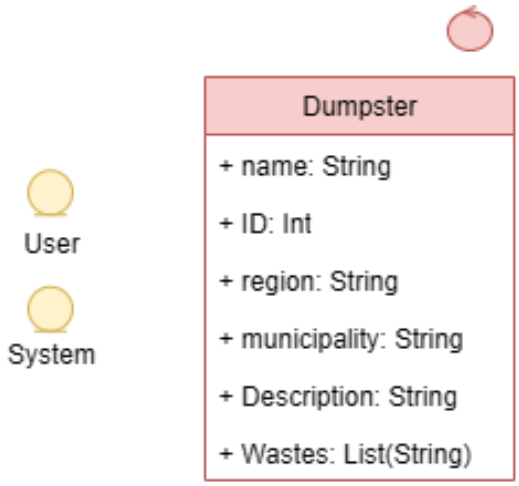


Il·lustració 16. Diagrama de classes del cas d'ús 8

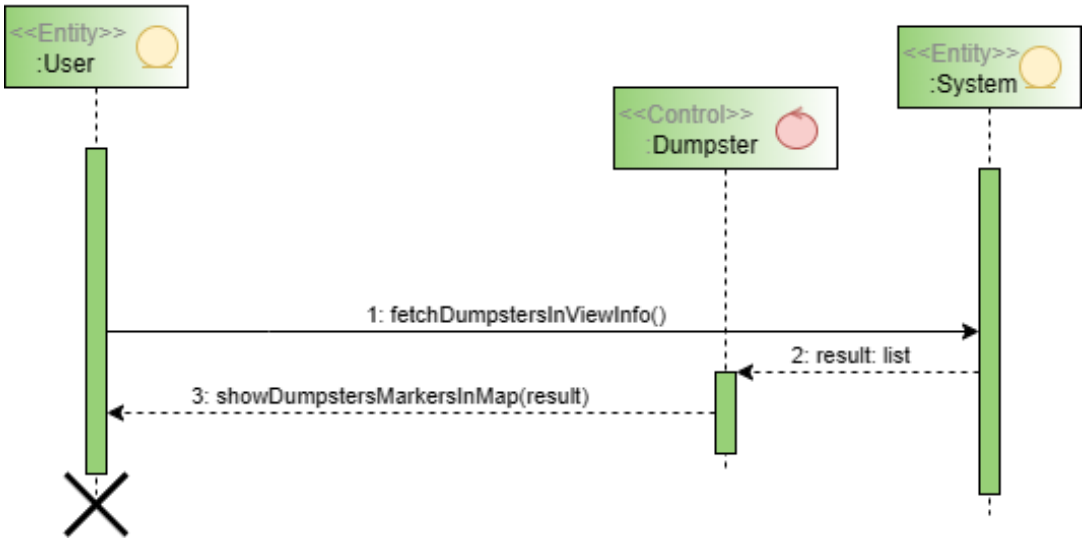


Il·lustració 17. Diagrama de seqüència del cas d´ús 8

RF-09. Consultar mapa d'abocadors

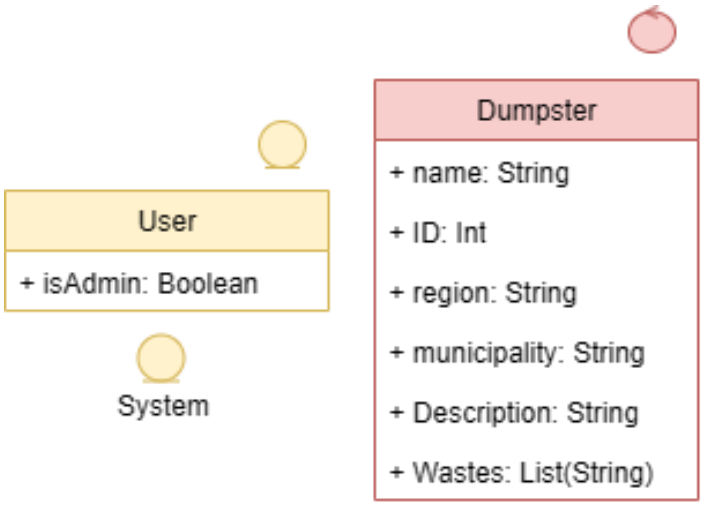


Il·lustració 18. Diagrama de classes del cas d'ús 9

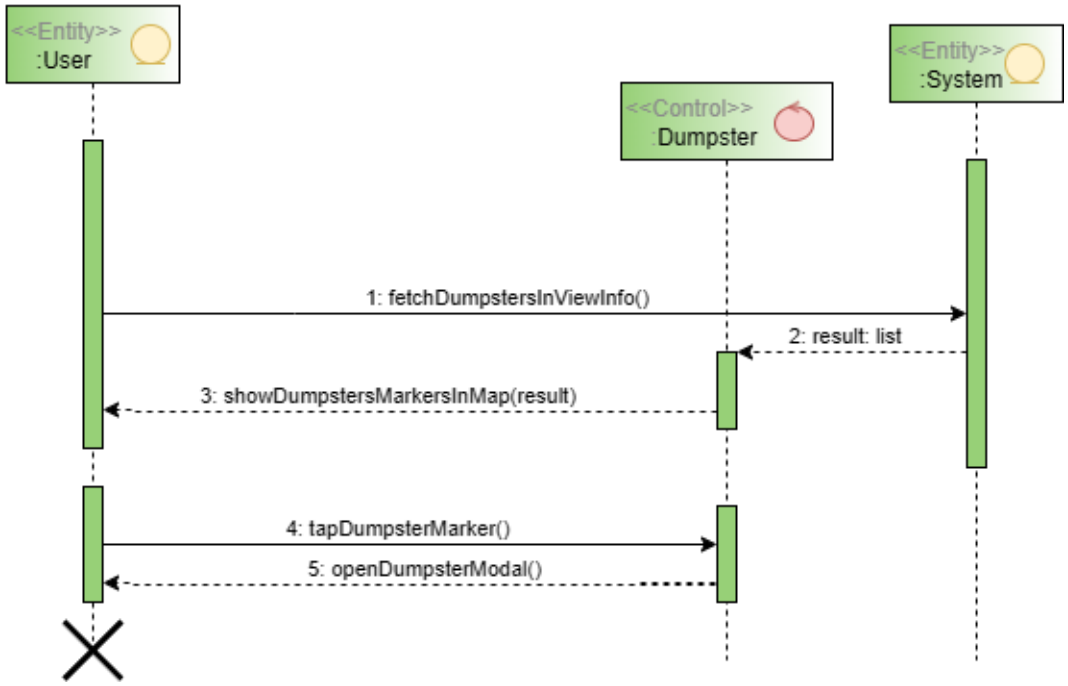


Il·lustració 19. Diagrama de seqüència del cas d´ús 9

RF-10. Consultar informació d'abocador al mapa

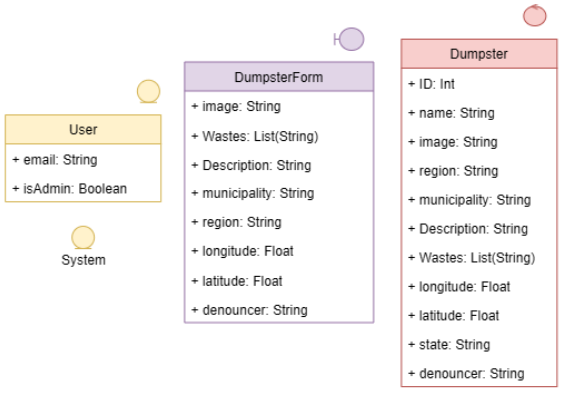


Il·lustració 20. Diagrama de classes del cas d'ús 10

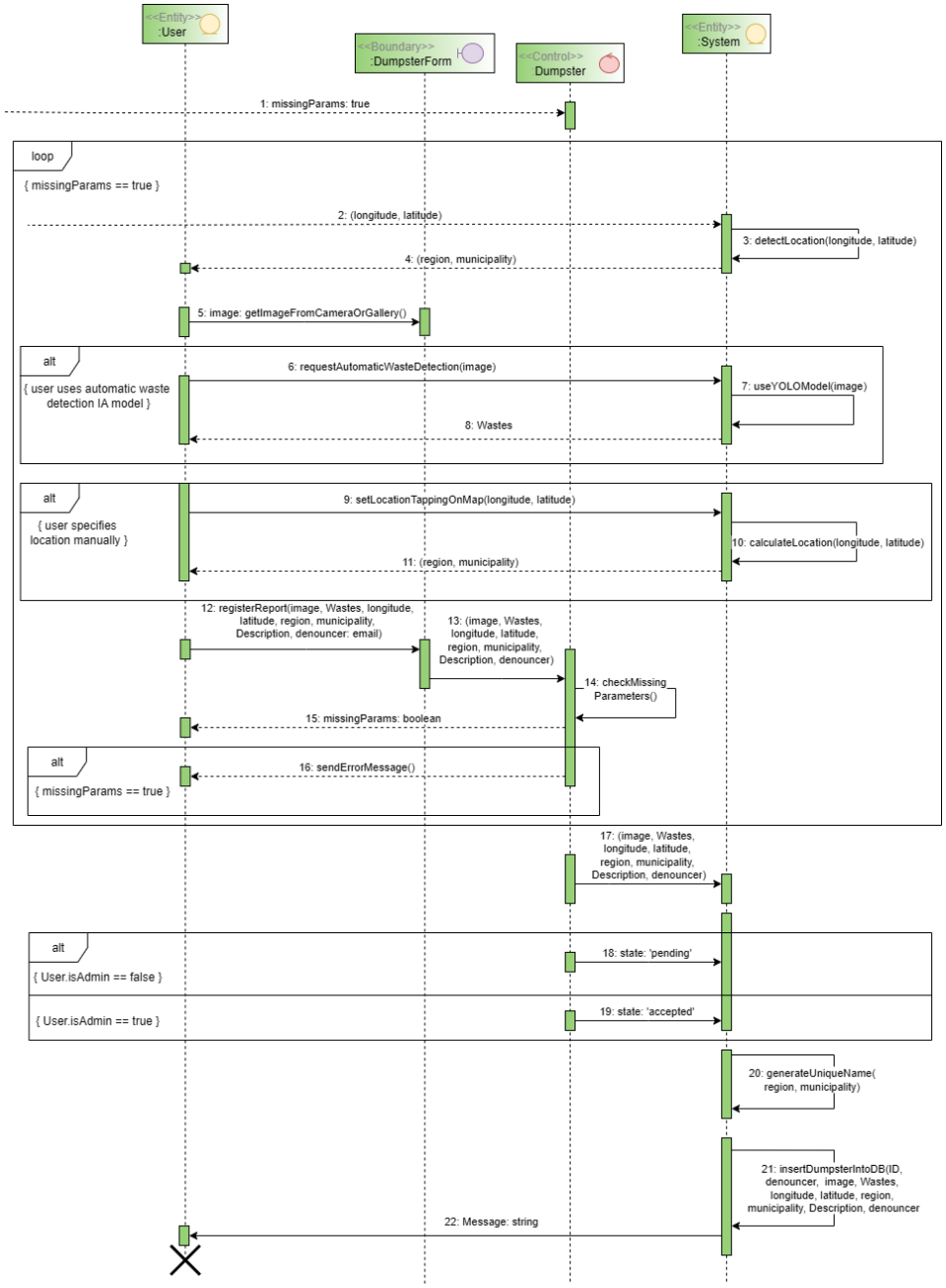


Il·lustració 21. Diagrama de seqüència del cas d´ús 10

RF-11. Enregistrar nova denúncia d'abocador

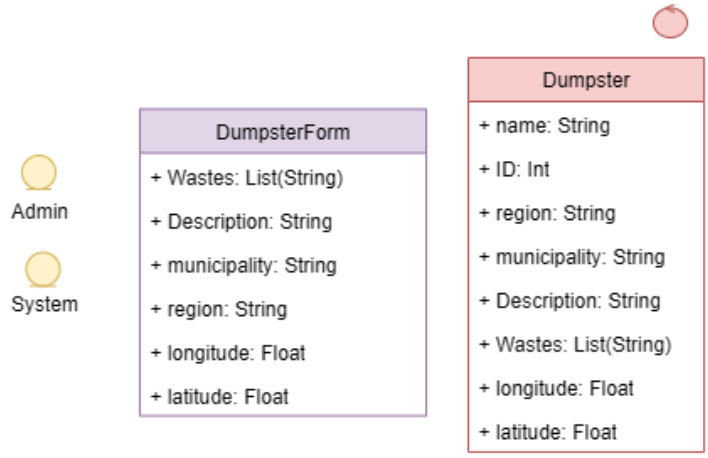


Il·lustració 22. Diagrama de classes del cas d'ús 11

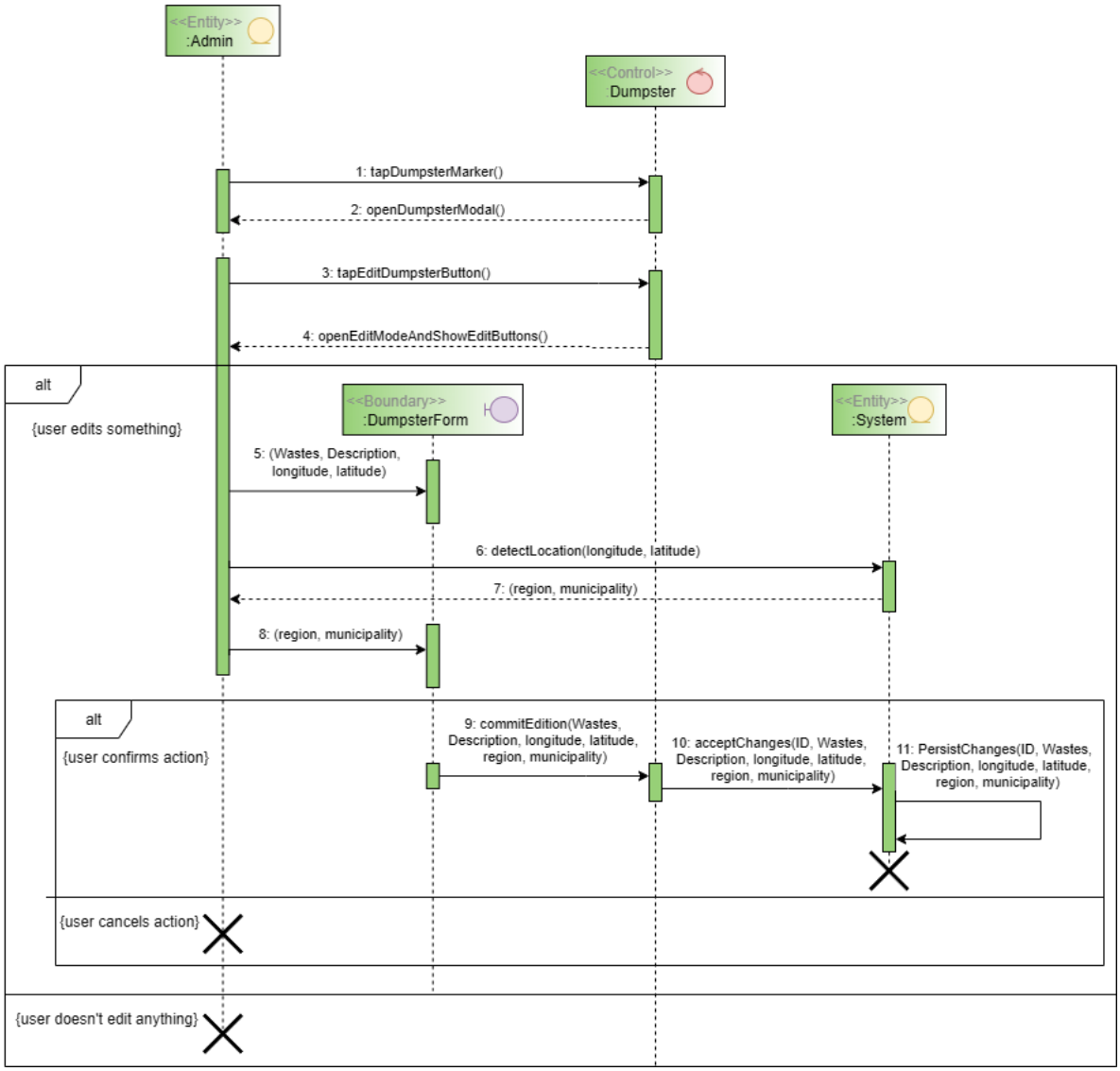


Il·lustració 23. Diagrama de seqüència del cas d´ús 11

RF-12*. Editar abocador

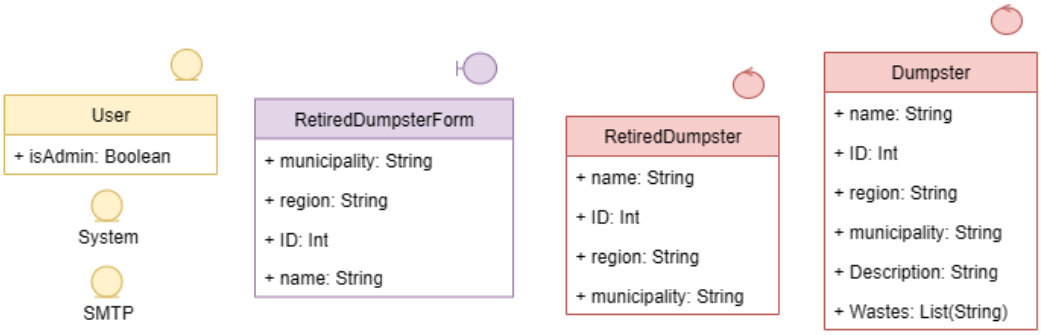


Il·lustració 24. Diagrama de classes del cas d'ús 12

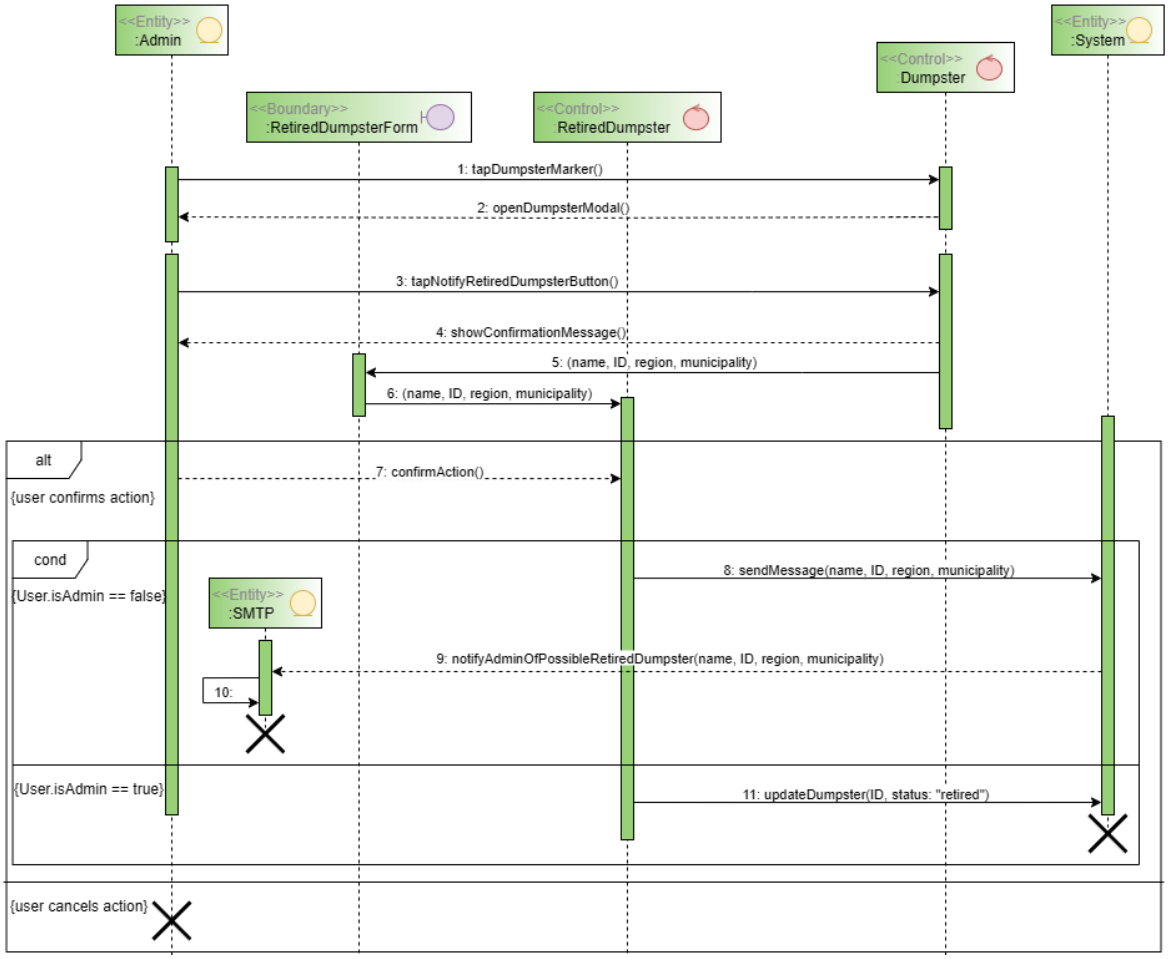


Il·lustració 25. Diagrama de seqüència del cas d´ús 12

RF-13. Retirar abocador



Il·lustració 26. Diagrama de classes del cas d'ús 13



Il·lustració 27. Diagrama de seqüència del cas d´ús 13

5 Disseny

En aquest apartat es presenta el procés de disseny del sistema, tot detallant les decisions preses i la seva justificació. L'objectiu és explicar no només **què** s'ha escollit (arquitectura, tecnologies i mòduls principals), sinó també **per què** s'han pres aquestes decisions, considerant criteris com escalabilitat, mantenibilitat, rendiment i adequació als objectius del projecte.

5.1 Arquitectura del Projecte

Aquest projecte segueix una arquitectura distribuïda de tres capes que separa clarament les responsabilitats del sistema. Aquesta decisió arquitectural segueix els criteris abans comentats així com la flexibilitat en el desenvolupament i desplegament de cada component de manera independent.

Per al sistema s'ha optat per una arquitectura client-servidor que es compon dels següents elements principals:

- Mòdul d'Intel·ligència Artificial: Model YOLOv11 integrat a l'API.
- Capa de Presentació: Aplicació mòbil desenvolupada amb React Native + Expo.
- Capa de Lògica de Negoci: API REST desenvolupada amb Python + FastAPI.
- Capa de Persistència: Base de dades PostgreSQL.

5.1.1 Mòdul d'Intel·ligència Artificial

Per al mòdul d'intel·ligència artificial es va escollir **YOLOv11 d'Ultralytics**, integrat amb la seva API de Python. Aquesta decisió es va prendre després d'analitzar alternatives en dues grans categories:

- **Models de Classificació d'Imatges**, com **ResNet** [14][15], **MobileNet** [16][18] o **EfficientNet** [18][22], útils per assignar una classe a tota la imatge però no poden detectar múltiples residus ni localitzar-los [20][21][22][23].
- **Models de Detecció d'Objectes**, com **YOLO**, **EfficientDet** [24][25], **SSD**[15] [26][27], **DETR**[16] [28] o R-CNN[17] com **Faster R-CNN** [29] que permeten identificar i localitzar múltiples residus dins d'una imatge mitjançant *bounding boxes*[18], ajustant-se millor a l'objectiu del projecte

[15] SSD: Single Shot multibox Detector

[16] DETR: Detection Transformer

[17] R-CNN: Region-based Convolutional Neural Network

[18] Bounding Box: Rectangle imaginari que envolta completament un objecte dins d'una imatge o escena gràfica, utilitzat per a la detecció, selecció o càlcul de col·lisions.

Per escollir el model final, es van comparar algunes de les arquitectures més rellevants tenint en compte criteris com el rendiment (mAP[19]@50) i la latència en GPU[20]:

MODEL	mAP@50	LATÈNCIA GPU
YOLOv5	0.58	~2.3ms
YOLOv8	0.62	~1.4ms
Faster R-CNN	0.41	~54ms
EfficientDet	0.47	~3.9ms

Taula 2. Comparació de diferents models de detecció d'objectes

La comparació de diferents arquitectures va evidenciar que Faster R-CNN era massa lent (~54 ms) [30] i que EfficientDet, tot i eficient en entorns amb recursos limitats, resultava menys precís [31]. YOLOv8 presentava un equilibri molt competitiu (mAP=0.62, latència ~1.4 ms) [32], però finalment, després de valorar les iteracions posteriors (YOLOv9, v10...),es va optar per YOLOv11, l'última versió estable en el moment del desenvolupament, que supera YOLOv8 en precisió i eficiència [8].

Els criteris decisius van ser:

1. **Requisits Funcionals del Projecte**: La detecció multi objecte i la localització espacial són clau per futures funcionalitats com el comptatge automàtic.
2. **Rendiment i Eficiència**: YOLOv11, gràcies a la seva arquitectura *single-shot*[21], pot processar imatges en temps real, crucial per a l'experiència d'usuari en una aplicació mòbil.
3. **Facilitat d'Integració**: API Python senzilla i compatible amb FastAPI que permet carregar el model amb una sola línia i utilitzar directament sobre imatges, simplificant el desenvolupament.
4. **Ecosistema i Documentació**: YOLOv11 compta amb suport per entrenament, validació, optimització d'hiperparàmetres (Ray Tune [33]), i exportació a diferents formats.

L'entrenament es va realitzar mitjançant *transfer learning* a partir del model yolov11x.pt, amb optimització d'hiperparàmetres i monitoratge de mètriques en GPU, assegurant un model adaptat a la detecció de residus en escenaris reals.

Unes setmanes després d'acabar l'entrenament del model v11 Ultralytics va llançar el model v12 [34] el qual segurament es faci servir per a futures millores d'aquest projecte.

[19] mAP: mean Average Precision

[20] GPU: Graphic Processing Unit

[21] Single-shot: Enfocament o tècnica que realitza una acció o càlcul en una única passada, sense necessitat de processos iteratius o múltiples etapes, habitualment per a detecció o reconeixement ràpid d'objectes.

5.1.2 Capa de Presentació

Per a la capa de presentació es va optar per desenvolupar una aplicació mòbil nativa utilitzant **React Native** amb el *framework* **Expo**. Aquesta decisió es va prendre després d'avaluar les alternatives disponibles i considerant els requisits específics del projecte.

Inicialment es va plantejar el dubte de si fer servir FlutterFlow o React Native per a desenvolupar l'aplicació mòbil. Aquesta decisió era crucial doncs condicionaria tot el procés de desenvolupament, les capacitats futures de l'aplicació i la corba d'aprenentatge necessària.

FlutterFlow és una plataforma de desenvolupament visual que permet crear aplicacions Flutter mitjançant una interfície de *drag-and-drop* Els seus principals avantatges inclouen un desenvolupament ràpid per a MVP[22], una corba d'aprenentatge suau sense necessitat de coneixements previs de Dart/Flutter, connexions predefinides amb serveis populars i la capacitat d'exportar a Flutter natiu si és necessari [35].

Tot i això, FlutterFlow presenta limitacions significatives per al tipus de projecte que es volia desenvolupar: la interfície drag-and-drop limita les optimitzacions i el control arquitectural, la implementació de funcionalitats personalitzades es veu dificultada, i la integració amb models propis d'intel·ligència artificial pot resultar complicada.

React Native en canvi, és un framework de desenvolupament mòbil que permet crear aplicacions natives utilitzant JavaScript i React, proporcionant control total sobre l'arquitectura i optimitzacions, i facilitant la implementació de funcionalitats personalitzades. Com és d'esperar, té una corba d'aprenentatge més complexa que FlutterFlow [35].

La decisió final va inclinar-se cap a React Native basant-se en els següents criteris prioritaris:

1. **Requisits Funcionals Específics**: L'aplicació requereix funcionalitats GPS, mapes interactius, compressió, manipulació i emmagatzematge d'imatges i la capacitat de poder desenvolupar *offline* sense haver de dependre d'un servei extern [36].
2. **Objectius Formatius**: El fet d'haver d'aprendre React Native aporta coneixements aplicables al món laboral i una comprensió real del desenvolupament mòbil.
3. **Escalabilitat i Mantenibilitat**: Es preveu un creixement ràpid en complexitat tecnològica i base d'usuaris, beneficiant-se del control total sobre codi i arquitectura.
4. **Integració amb l'ecosistema**: React Native permet una integració eficient amb l'API desenvolupada amb FastAPI, complementat per Expo per facilitar el desenvolupament i llançament.

[22] MVP: Minimum Viable Product

5.1.3 Capa de Lògica de Negoci

La capa de lògica de negoci desenvolupa una API REST[23] utilitzant **Python** amb el framework **FastAPI**. Aquesta decisió es va prendre després d'avaluar què aniria millor per al projecte, **Node.js + Express** o **Python + FastAPI**. I considerant els requisits específics del projecte, es va prendre la decisió de fer servir **Python + FastAPI**.

El principal motiu per a prendre aquesta decisió va ser el fet de que s'havia d'integrar d'alguna manera el model de IA, pensat per a ser executat amb Python, cosa que dificultaria el desenvolupament d'una API feta amb Node.js + Express, però tot i ser el motiu principal, hi havia moltes altres raons per a escollir Python + FastAPI:

1. **Rendiment Superior**: FastAPI ofereix un dels millors rendiments entre els frameworks de Python, comparable amb frameworks de Node.js, gràcies al seu suport natiu per a programació asíncrona (ideal per a enviar imatges al *frontend*).
2. **Documentació Automàtica**: Genera automàticament documentació interactiva OpenAPI/Swagger basada en les anotacions de tipus de Python, facilitant el desenvolupament i testing de l'API.
3. **Validació de Dades Integrada**: Es pot utilitzar Pydantic per a la validació automàtica de dades d'entrada i sortida, reduint significativament el codi *boilerplate*[24] i millorant la robustesa [37].
4. **Suport Asíncron Natiu**: Permet gestionar múltiples peticions concurrents de manera eficient, crucial per a una aplicació que pot tenir molts usuaris simultanis com podria arribar a ser el cas d'**AbocadorsBP**.
5. **Integració amb l'Ecosistema Python**: Facilita la integració directa amb llibreries de ML[25] i processament d'imatges necessàries per al model YOLO.

També cal destacar que es va contemplar l'opció de fer servir Firebase/Firestore o Supabase com a lògica de negoci i DB[26] al mateix temps però aquestes opcions feien perdre control sobre el backend i per tant es van descartar.

5.1.4 Capa de Persistència

Per a la capa de persistència del sistema es va optar per utilitzar una base de dades SQL[27], ja que totes les dades seran estructurades, específicament **PostgreSQL,** com a sistema

---

[23] API REST: conjunt d'interfícies de programació que permeten la comunicació entre sistemes utilitzant el protocol HTTP i els principis de representational state transfer (REST), facilitant l'intercanvi de dades de manera estructurada i escalable.

[24] Boilerplate: conjunt de codi o contingut reutilitzable que serveix de base estructural per a un projecte o component, reduint la necessitat d'escriure elements repetitius des de zero.

[25] ML: Machine Learning

[26] DB: Data Base

[27] SQL: Structured Query Language

de gestió de base de dades relacional. Aquesta decisió es va prendre després d'avaluar diverses alternatives considerant els requisits específics del projecte.

Les opcions que es van considerar eren **MySQL** i **PostgreSQL**, sent la segona l'opció escollida pels següents motius:

1. **Suport Geoespacial Natiu:** PostgreSQL inclou l'extensió **PostGIS**, que proporciona funcionalitats avançades per a l'emmagatzematge i consulta de dades geoespacials [38]. Aquesta extensió no s'utilitza actualment al projecte però és útil que estigui disponible per si en un futur es vol migrar i fer-la servir ja que ofereix molts beneficis que podrien beneficiar a l'aplicació:
    - Permet emmagatzemar coordenades de latitud i longitud de manera eficient
    - Ofereix funcions optimitzades per a consultes espacials (cerques per proximitat, dins de límits geogràfics)
    - Suporta índexs espacials per a consultes ràpides sobre grans volums de dades geogràfiques
2. **Robustesa i Fiabilitat:** PostgreSQL compta amb un sistema ACID[28] complet que garanteix la consistència de les dades, està pensat per a tenir temps de resposta competitius i capacitat de manejar càrregues de treball intensives de manera estable.

    El seu sistema de recuperació i còpies de seguretat és madur i fiable [39].
3. **Ecosistema Python:** PostgreSQL compta amb una excel·lent integració amb SQLAlchemy, l'ORM[29] utilitzat en la API i una comunitat activa i documentació extensa.

5.1.5 Diagrama Conceptual

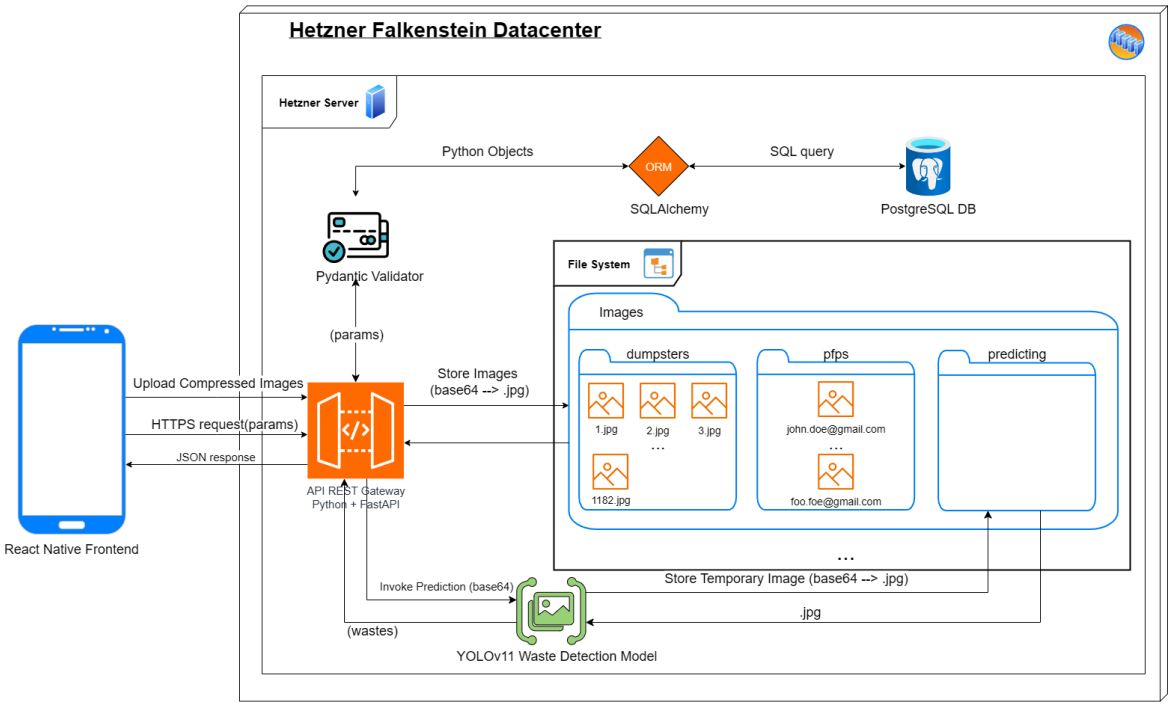
Seguidament es mostra un diagrama conceptual que facilita la visualització de l'arquitectura proposada.

Cal destacar que el servidor, és a dir, la part de l'arquitectura que engloba la capa de lògica de negoci, la capa de persistència i el mòdul d'intel·ligència artificial, està allotjat a la plataforma Hetzner, concretament al *datacenter*[30] de Falkenstein, per tant, ens referim al "Servidor Hetzner" quan parlem del conjunt d'aquestes tres parts de l'arquitectura:

[28] ACID: Atomicity Consistency Isolation Durability

[29] OMR: Object-Relational Mapping

[30] Datacenter: instal·lació física que allotja sistemes informàtics, servidors i equips de xarxa, dissenyada per emmagatzemar, processar i gestionar grans volums de dades de manera segura i fiable.



Il·lustració 28. Diagrama conceptual de l'arquitectura del projecte

5.2 Disseny de l'Entrenament i la Integració del Model Classificador de Residus

En aquest apartat es detalla el procés d'entrenament, validació i integració del model de detecció de residus, així com les decisions preses durant el desenvolupament.

5.2.1 Introducció a YOLO

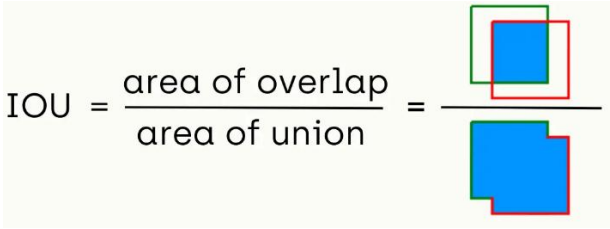
YOLO és un model de detecció d'objectes en una sola passada (o *single-shot*): en lloc de seguir el procés de dues etapes (primer seleccionar regions candidates i després classificar-les, com Faster R-CNN), YOLO aplica una única xarxa neuronal convolucional a tota la imatge per predir directament totes les caixes delimitadores i classes al mateix temps [40][41]. En paraules de Redmon et al., «YOLO processa la imatge només una vegada» i genera ràpidament totes les deteccions en una sola passada [40], cosa que li atorga una velocitat d'inferència molt elevada. Per exemple, YOLOv3 pot proporcionar fins a 8 cops més imatges per segon que Faster R-CNN tot mantenint una precisió comparable [41]. Aquesta eficiència és clau en aplicacions en temps real (vigilància, vehicles autònoms, drons, etc.), on la latència és crítica.

El funcionament intern de YOLO es pot descriure així: primer es redimensiona cada imatge d'entrada a una mida fixa (per exemple, 448×448 píxels) i es divideix en una graella de mida S×S (al paper oficial es fa servir S = 7). Cada cel·la de la graella "s'encarrega" dels objectes el centre dels quals cau dins seu. Concretament, cada cel·la prediu *B* caixes delimitadores (bounding boxes, al paper oficial es fa servir B = 2). Cada una d'aquestes caixes consisteix de 5 prediccions: les coordenades **x, y, w, h** i la puntuació de confiança. Les coordenades (x, y) representen el centre de la caixa delimitant en relació amb la cel·la mentre que **h**(eight) i **w**(idth) es prediuen relatives a la imatge. Finalment la puntuació de confiança indica tant la probabilitat que la caixa conté un objecte com la precisió de la predicció de la caixa [43].

La puntuació de confiança es defineix com la probabilitat d'un objecte multiplicada per la IoU [31]. És a dir, la precisió de la predicció de la *bounding box*:

$$(1)\ P(\text{Object}) * IoU^{truth}_{pred}$$

La IoU s'entén com el solapament entre la *bounding box* que s'ha predit i la *bounding box* real, també anomenada *ground truth bounding box*:



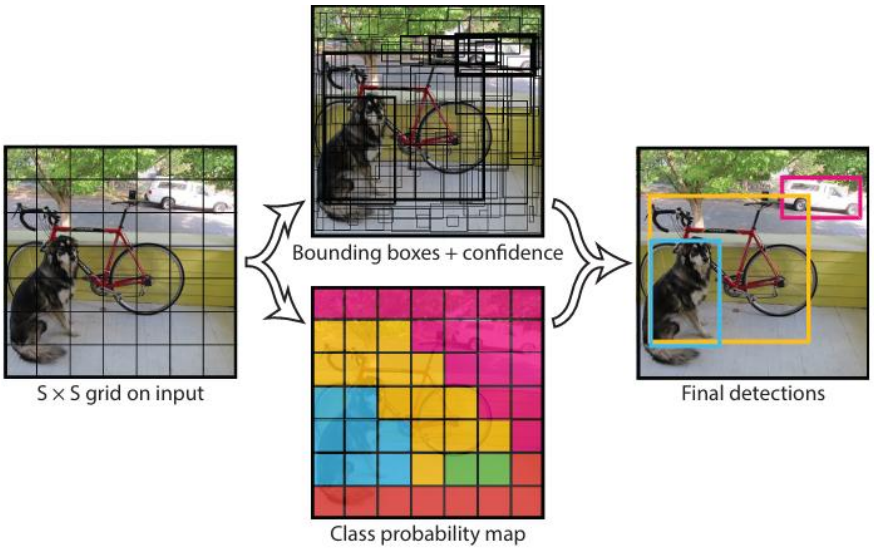
Il·lustració 29. Representació visual del càlcul de la IoU

A més, cada cel·la genera *C* probabilitats condicionals de classe ($P(c_i)$ per a cada classe c_i) assumint que hi ha un objecte present. La probabilitat final que en la caixa aparegui l'objecte de la classe c_i es calcula multiplicant la confiança de la caixa pel $P(c_i)$ corresponent:

$$(2)\ P(c_i) * IoU^{truth}_{pred}$$

D'aquesta manera, en una sola passada la xarxa obté, per a cada cel·la, diverses caixes i probabilitats de classe [43]. Aquestes prediccions es codifiquen com un tensor de les següents mides:

$$(3)\ S \times S \times (B * 5 + C)$$



Il·lustració 30. Funcionament intern del model YOLO per etapes

L'arquitectura del model es basa en una xarxa neuronal profunda constituïda per **24 capes convolucionals** (encarregades d'extreure característiques com textures, vores, formes, patrons complexos i altres de les imatges) seguides de **dues capes completament**

[31] IoU: Intersection Over Union

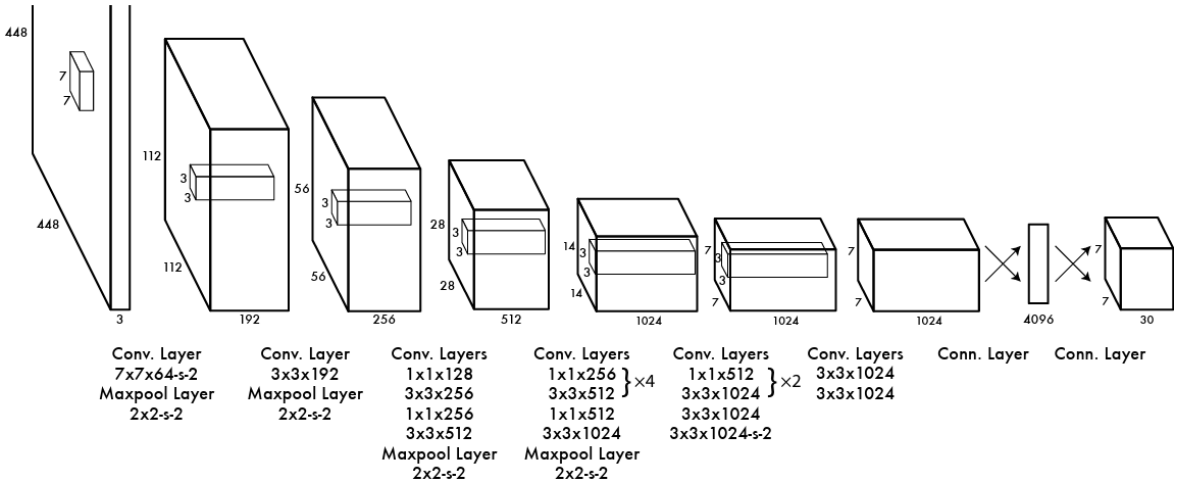
connectades (FC[32]) (encarregades de combinar aquestes característiques per proporcionar la sortida final del tensor.

Per optimitzar el rendiment i reduir la complexitat computacional, el model introdueix **capes convolucionals de mida 1×1** de forma alternada. Aquestes capes no analitzen la informació espacial, sinó que actuen reduint la dimensionalitat del nombre de canals en els mapes de característiques provinents de capes anteriors. D'aquesta manera, es manté la informació rellevant i alhora es minimitza el cost de càlcul, la qual cosa permet aprofundir més en l'arquitectura sense comprometre'n l'eficiència.

Un altre aspecte fonamental és l'estratègia d'entrenament. Les capes convolucionals del model es **preentrenen en la base de dades ImageNet** per a la tasca de classificació d'imatges, amb una resolució d'entrada de 224×224 píxels. Aquesta fase de preentrenament permet que el model aprengui representacions visuals generals (vores, textures, formes i patrons), que posteriorment es poden reutilitzar i adaptar a la tasca específica de detecció d'objectes.

Finalment, en la fase de detecció, la resolució d'entrada de les imatges es **duplica fins a 448×448** píxels. Aquest increment de resolució proporciona més detall espacial i millora la localització dels objectes dins la imatge, especialment en el cas d'objectes petits, la qual cosa és clau per a obtenir prediccions de bounding boxes més precises [43].

A continuació es proporciona un diagrama amb l'arquitectura del model YOLO bàsic:



Il·lustració 31. Arquitectura del model YOLOv1

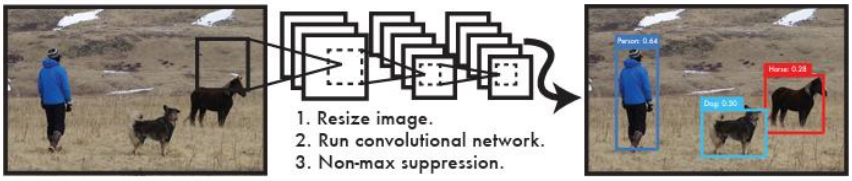
Aquest enfocament global unificat fa que la detecció sigui extremadament ràpida i coherent, però requereix postprocessament per filtrar solapaments. En concret, un cop la xarxa retorna totes les caixes, s'aplica habitualment una **supressió de màxims locals** (NMS[33]) per eliminar caixes redundants que es solapen molt [41][43].

[32] FC: Fully Connected

[33] NMS: Non-Maximum Suppression

NMS ordena totes les caixes segons la seva *confidence score* (anomenada anteriorment "puntuació de confiança") i escull la que tingui la puntuació més alta per marcar-la com a predicció final. Seguidament, compara aquesta caixa amb totes les altres mitjançant la seva IoU i si aquesta sobrepassa un llindar (p.e. 0.5), vol dir que se solapen massa i que probablement representen el mateix objecte, per tant es descarten. Aquest procés es repeteix amb les caixes que queden de manera que al final, només sobreviuen les caixes més representatives i es redueix el soroll de deteccions duplicades.

En resum, l'arquitectura general de YOLO es basa en una xarxa neuronal convolucional única que mira tota la imatge i prediu simultàniament caixes delimitadores i probabilitats de classe. La imatge es divideix en graella S×S, cada cel·la retorna B caixes amb una confiança i probabilitats de classe, i després s'aplica NMS per afinar els resultats.



Il·lustració 32. Fases del sistema de detecció YOLO

5.2.1.1 Evolució del Model YOLO

Com hem vist, la primera versió de YOLO (v1) predeia les coordenades absolutes de les caixes i les classes directament amb capes FC. A partir de YOLOv2, el model va adoptar la tècnica d'*anchor boxes*, caixes predefinides amb diferents *aspect ratios*[34] i escales que serveixen de base per a la regressió de coordenades [41]. L'ús d'ancoratges ajuda a manejar objectes de formes i mides diverses sense haver de predir-les des de zero, millorant la precisió.

En versions posteriors (v3, v4, v5, etc.), es van introduir millores com xarxes d'extracció de característiques més profundes (Darknet-53 a YOLOv3, CSPNet a v4), pèrdues ajustades (GIoU[35], CIoU[36]) i tècniques d'augment de dades (mosaic, mixup, etc.) per reduir l'error de localització i detecció d'objectes petits [41].

Posteriorment amb la introducció de YOLOv8 es va deixar enrere el backbone basat en Darknet i es va passar a un marc desenvolupat en PyTorch, la qual cosa va facilitar molt l'ús, la personalització i la integració en entorns de recerca i producció.

YOLOv8 també introdueix un *head*[37] d'anotació per a detecció *anchor-free*, en el qual es deixen enrere les *anchor boxes* fixes. Aquesta estratègia millora la detecció d'objectes petits i variats, redueix la dependència d'hiperparàmetres predefinits i simplifica l'entrenament. Com a valor afegit, YOLOv8 incorpora millores importants en l'estratègia de

[34] Aspect ratio: Proporció entre l'amplada i l'alçada d'una imatge, pantalla o element gràfic, que determina la seva forma i influència en la presentació visual del contingut.

[35] GIoU: Generalized Intersection over Union

[36] CIoU: Complete Intersection over Union

[37] Head (context de Xarxes neuronals): Capa o secció final d'una xarxa neuronal que produeix les prediccions de detecció d'objectes, incloent-hi les coordenades del bounding box, la classe i la confiança de la detecció.

pèrdua, amb l'ús de variants avançades de IoU i en les tècniques d'augment de dades (mosaic, mixup, copy-paste, etc.), que permeten obtenir models més robustos davant situacions reals. El resultat és un model més ràpid, més precís i més flexible, que consolida YOLO com un dels estàndards per a tasques de visió per computador. YOLOv8 es converteix doncs en la base per a futurs models com YOLOv9, YOLOv10 o el que es fa servir en aquest projecte, YOLOv11 que en el futur migrarà a YOLOv12.

5.2.2 El Model Utilitzat

El model que s'utilitza en aquest projecte és YOLOv11 (també anomenat Ultralytics YOLO11). En comparació amb YOLOv8, destaca per la seva precisió i eficiència computacional. Totes les variants de YOLOv11 superen en *mAP* els models equivalents amb menys paràmetres i FLOP[38]s.

A continuació es presenta una comparativa quantitativa entre algunes variants de YOLOv11 i les corresponents de YOLOv8 a resolució 640×640, mostrant diferències en precisió (*mAP@*50-95), nombre de paràmetres, operacions de punt flotant (FLOPs) i temps d'inferència en GPU i CPU[39] [44]:

Model	**mAP@50-95 (%)**	**Paràmetres (M)**	**FLOPs (G)**	**Temps GPU (ms)**	**Temps CPU (ms)**
YOLOv11n	39.5	2.6	6.5	1.5	56.1
YOLOv11s	47.0	9.4	21.5	2.5	90.0
YOLOv11m	51.5	20.1	68.0	4.7	183.2
YOLOv11l	53.5	25.3	86.9	6.2	238.6
YOLOv11xl	54.7	56.9	194.9	11.3	462.8
YOLOv8n	37.3	3.2	8.7	3.2	80.4
YOLOv8s	44.9	11.2	28.6	11.2	128.4
YOLOv8m	50.2	25.9	78.9	25.9	234.7
YOLOv8l	52.9	43.7	165.2	43.7	375.2
YOLOv8xl	53.9	68.2	257.8	68.2	479.1

Taula 3. Comparativa de rendiment entre variants de YOLOv11 i YOLOv8 a resolució 640×640

Aquestes dades reflecteixen no només un increment de precisió, sinó també una **eficiència computacional superior**. YOLOv11 aconsegueix un *mAP* lleugerament més alt

[38] FLOP: Floating Point Operation

[39] CPU: Central Processing Unit

amb menys FLOPs gràcies a optimitzacions arquitectòniques com blocs del *backbone*[40] i del *neck*[41] més eficients, reducció de paràmetres redundants i millor gestió de les prediccions de *bounding boxes* amb pèrdues com la CIoU. L'estructura modular permet un processament paral·lel més ràpid, disminuint la latència d'inferència, especialment en CPU.

Per aconseguir aquestes millores, YOLOv11 introdueix canvis respecte a versions anteriors, com una major robustesa davant la variabilitat de les imatges, optimització de la pèrdua i un millor rendiment en la detecció de múltiples residus per imatge, proporcionant tant classificació com localització espacial. Tot i basar-se en l'arquitectura de YOLOv8 (*backbone – neck – head*), YOLOv11 incorpora nous mòduls i ajustos que afinen l'extracció de característiques visuals [8][45] i introdueixen funcionalitats addicionals com estimació de posat (*pose*), segmentació i detecció amb caixes rotades, les quals no s'utilitzen en aquest projecte però subratllen la seva superioritat arquitectònica.

5.2.3 Datasets Utilitzats

Inicialment es va considerar l'ús del conjunt de dades **Trashnet**, creat per Gary Thung i Mindy Yang [46]. Tot i això, aquest conjunt de dades està principalment orientat a la classificació d'imatges, ja que en la majoria de casos els objectes ocupen tota la imatge. Aquesta característica dificulta l'aprenentatge del model per a tasques de detecció amb caixes delimitadores. A més, la versió disponible a Kaggle es presentava amb les imatges organitzades en carpetes segons la seva categoria, però **no seguia l'estructura necessària per entrenar models YOLO**, que requereix que per a cada imatge (nom_imatge.jpg) hi hagi un fitxer de text associat (nom_imatge.txt) amb les anotacions corresponents [47][48]. Per aquests motius, Trashnet es va descartar.

En canvi, es va decidir explorar el conjunt de dades del projecte TACO[42], que, a diferència de Trashnet, està orientat a la detecció d'objectes (més específicament a la segmentació de residus). TACO proporciona imatges reals de residus etiquetades en format COCO[43]. Tot i que aquest format no és compatible directament amb YOLO, la seva conversió és relativament senzilla [49][50][51].

Tanmateix, TACO presenta una limitació important, només **disposa de 1.500 imatges amb anotacions oficials**, és a dir, anotacions revisades i de qualitat, amb *bounding boxes* que s'ajusten correctament als objectes. El conjunt complet arriba a 5.236 imatges, de les quals 3.736 compten amb anotacions comunitàries que no són oficials i que poden contenir errors, imprecisions o caixes delimitadores mal ajustades [52][53]. Això redueix de manera significativa la quantitat de dades fiables disponibles per a l'entrenament.

Tenint en compte que només una part reduïda de les dades és plenament utilitzable, i que durant l'entrenament cal reservar aproximadament un **70–75% per a entrenament i la**

[40] Backbone: part de la xarxa neuronal que s'encarrega d'extreure característiques rellevants de la imatge d'entrada, generant mapes de característiques que resumeixen la informació visual.

[41] Neck: capa intermèdia que combina i refina les característiques extretes pel backbone, preparant-les per a la capa head i millorant la detecció d'objectes a diferents escales.

[42] TACO: Trash Annotations in Context

[43] COCO: Common Objects in Context

resta per a validació i test, es va concloure que el *dataset*[44] de TACO, per si sol, era insuficient. Per aquest motiu, es va decidir combinar-lo amb altres conjunts de dades de residus provinents de RoboFlow, que inclouen des de residus quotidians [54] fins a runa d'obra[55][56] i mobles, alguns d'ells trencats [57][58][59], entre d'altres. En aquest procés es va procurar seleccionar conjunts de dades que s'apropessin a la realitat dels abocadors il·legals que es volen denunciar, ampliant així la diversitat i el volum d'exemples disponibles, fent que el conjunt final estigués format per 14.974 imatges, 10 cops més que solament usant TACO. La decisió de recórrer principalment a RoboFlow es deu al fet que aquesta plataforma permet exportar els conjunts de dades en diversos formats d'anotació, entre els quals es troba el format YOLO, la qual cosa va simplificar enormement la integració amb el model.

Finalment, per tal d'assegurar la disponibilitat i evitar possibles problemes de latència durant l'entrenament, totes les imatges es van descarregar localment. Atès que el model emprat requereix anotacions en format YOLO, es va dur a terme la conversió de les anotacions originals del format COCO al format YOLO, adequat per a Ultralytics.

5.2.4 Classes de Residus

Com s'ha comentat anteriorment, es va partir del conjunt de dades TACO, el qual inclou fins a 60 classes diferents de residus. Tot i així, aquestes categories presentaven un nivell de granularitat excessiu i poc adequat per als objectius del projecte, ja que incloïen elements amb poca rellevància en el context dels abocadors il·legals, com ara les llengüetes de llaunes (*Pop tab*) o les burilles de cigarreta (*Cigarette*).

Per tal d'augmentar la robustesa del model i simplificar la seva aplicabilitat pràctica, es va optar per reestructurar les classes originals en un conjunt reduït i més general. Concretament, les 60 categories de TACO es van reagrupar en sis classes principals, segons tipus general de residu. D'aquesta manera, es va aconseguir un **equilibri entre granularitat i simplicitat**, la qual cosa permet que el model sigui més eficient en entorns reals. Aquesta reestructuració queda reflexada a la següent taula:

CLASSES FINALS	CLASSES TACO	
Metal	• Aluminium foil • Battery • Aluminium blister pack • Metal bottle cap • Metal lid	• Food Can • Aerosol • Drink can • Scrap metal
Plastic	- Other plastic bottle - Clear plastic bottle - Plastic lid - Six pack rings - Spread tub - Tupperware	- Crisp packet - Plastic film - Disposable plastic cup - Other plastic cup - Plastic bottle cap - Carded blister pack

[44] Data set: Conjunt de dades

	- Disposable food container - Other plastic container - Squeezable tube - Garbage bag - Other plastic wrapper - Single-use carrier bag - Polypropylene bag	- Plastic gloves - Plastic utensils - Plastic straw - Other plastic - Foam cup - Foam food container - Styrofoam piece
Glass	- Glass bottle - Broken glass - Glass cup - Glass jar	
Paper & Cardboard	- Toilet tube - Other carton - Egg carton - Drink carton - Corrugated carton - Meal carton - Pizza box - Paper cup	- Paper bag - Plastified paper bag - Magazine paper - Tissues - Wrapping paper - Normal paper - Paper Straw
Organic Waste	- Food Waste	
Other/Miscellaneous	- Rope & strings - Shoe - Unlabeled litter	

Taula 4. Reestructuració de classes per a l'entrenament del model YOLO

A més, es va introduir una setena classe específica per a runa d'obra (*Rubble*) i una vuitena classe específica per a mobles (*Furniture*), ja que aquests residus apareixen amb freqüència en els abocadors il·legals i resulten rellevants per a la detecció automàtica.

Finalment, les classes utilitzades en l'entrenament del model van quedar definides de la següent manera (es relaciona cada classe amb el seu ID usat per a l'entrenament):

- Metal - 0
- Plastic - 1
- Glass - 2
- Paper & Cardboard - 3
- Rubble - 4
- Organic Waste - 5
- Other/Miscellaneous - 6
- Furniture - 7

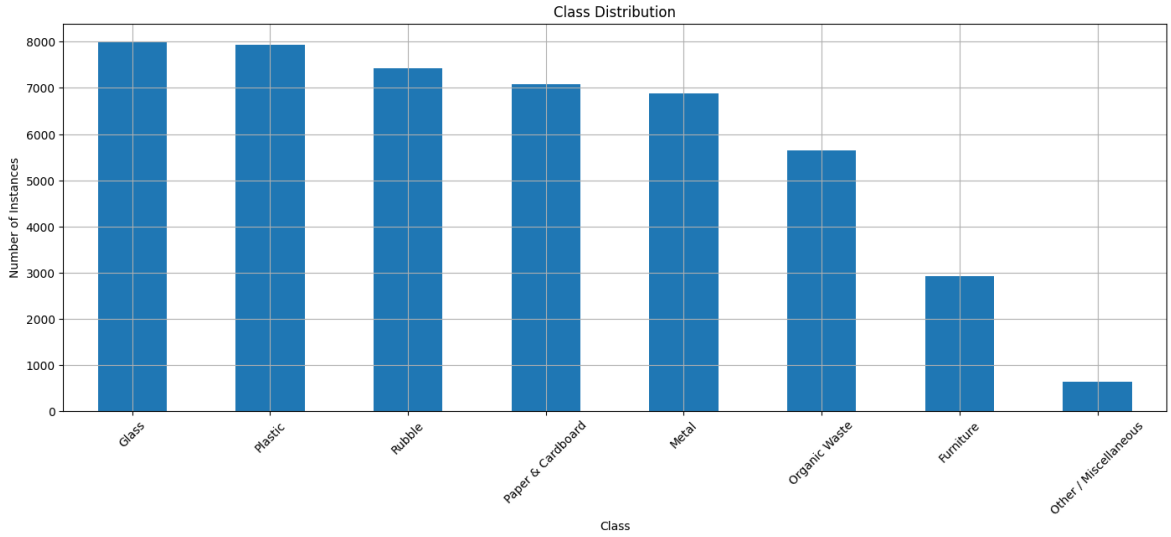
Aquesta reducció de classes permet disposar d'un model més **adaptat a la realitat dels abocadors il·legals de la zona**, alhora que manté una classificació prou detallada per a la seva posterior utilització en tasques de monitoratge i denúncia.

5.2.4.1 Distribució d'Instàncies per Classe

Després d'establir les classes finals, es va quantificar el nombre d'instàncies disponibles per a cada categoria dins del *dataset* reestructurat. Això permet conèixer la distribució de dades i identificar possibles desequilibris que podrien afectar l'entrenament del model. La següent taula i gràfic mostren el nombre d'instàncies per classe:

CLASSE	INSTÀNCIES
Glass	7.992
Plastic	7.928
Rubble	7.423
Paper & Cardboard	7.089
Metal	6.880
Organic Waste	5.646
Furniture	2.927
Other/Miscellaneous	633

Taula 5. Recompte d'instàncies per classe de residu



Il·lustració 33. Distribució d'instàncies per classe de residu

A partir d'aquesta informació, es pot observar que certes classes, com *Furniture* i *Other / Miscellaneous*, presenten un nombre d'instàncies considerablement menor que les altres, la qual cosa pot requerir tècniques d'augment de dades o estratègies d'entrenament específiques per equilibrar el model [60], tot i que aquestes opcions s'exploraran en un futur i no dins el marc d'aquest projecte.

A causa d'aquest desequilibri en el nombre d'instàncies, és previsible que el model presenti un **rendiment inferior** en la detecció de determinades categories, especialment aquelles amb menys exemples [60]. Tot i això, aquest risc es considera assumible en la fase inicial del projecte. De cara al futur, es preveu reentrenar el model amb dades més específiques, provinents de les aportacions dels usuaris a través de l'aplicació, la qual cosa permetrà millorar progressivament la seva capacitat de detecció en escenaris reals.

5.2.5 Procés d'Entrenament

Per al desenvolupament del model, s'ha dissenyat un procés d'entrenament que combina diverses estratègies per maximitzar el rendiment i la generalització.

En primer lloc, s'ha optat per l'estratègia de *transfer learning*, partint del model preentrenat yolov11x.pt, aprofitant pesos preentrenats en conjunts de dades grans per tal d'accelerar la convergència i millorar la capacitat del model per extreure característiques rellevants [61]. Seguidament s'ha procedit a fer *fine-tuning*[45] entrenant el model sobre el data set comentat anteriorment [62].

Per augmentar la variabilitat del conjunt d'entrenament i reduir el risc d'overfitting, l'entorn d'Ultralytics integra de manera nativa la llibreria Albumentations, especialitzada en *data augmentation*. Aquesta eina aplica transformacions com rotacions, canvis d'escala, ajustos de brillantor i contrast, o voltes horitzontals, entre d'altres. L'ús d'aquestes tècniques no només incrementa la robustesa i capacitat de generalització del model davant variacions en les imatges reals, sinó que també contribueix a mitigar parcialment el problema de **desequilibri de classes** present al conjunt de dades, ja que les transformacions generen mostres addicionals per a categories menys representatives [63].

La divisió del conjunt de dades s'ha realitzat en tres parts: entrenament (train), validació (val) i test. Aproximadament, el 70% de les dades s'han destinat a entrenament (10.481 imatges), el 15% a validació (2.246 imatges) i el 15% restant a test (2.246 imatges). Aquesta distribució permet disposar d'un conjunt suficient per entrenar el model, validar-ne el comportament durant l'entrenament i, finalment, avaluar-ne el rendiment de manera objectiva. La separació s'ha fet de manera aleatòria. Malgrat que no és l'opció més òptima a causa del possible desequilibri de classes, els resultats obtinguts han estat satisfactoris. En un futur, es plantejarà implementar una separació estratificada per combatre aquest desequilibri.

En segon lloc, s'ha dut a terme un procés d'ajust d'hiperparàmetres (*hyperparameter tuning*) mitjançant el *wrapper*[46] de Ray Tune integrat a l'API d'Ultralytics. Aquesta estratègia permet explorar diferents combinacions de valors per als hiperparàmetres clau i seleccionar aquells que **maximitzen el rendiment del model**, millorant així tant la convergència com la seva capacitat de generalització [64][65][66].

Pel que fa a l'optimitzador, inicialment es va considerar Adam[47] però finalment es va escollir SGD[48], atès que és un algoritme d'optimització àmpliament utilitzat en tasques de

[45] Fine-tuning: Procés d'ajustar els pesos d'un model preentrenat sobre un conjunt de dades específic, per adaptar-lo millor a una tasca concreta i optimitzar la seva precisió.

[46] Wrapper: capa o interfície que envolta una biblioteca, eina o funcionalitat existent, facilitant la seva integració i ús dins d'un entorn o flux de treball específic.

[47] Adam: Adaptive Moment Estimation

[48] SGD: Stochastic Gradient Descent

visió per computador amb conjunts de dades grans i que, a diferència d'Adam, sol proporcionar **millor capacitat de generalització** a llarg termini [67][68].

Donades les limitacions de recursos disponibles (GPU NVIDIA RTX 3060, 12GB), l'ajust d'hiperparàmetres es va centrar en tres variables principals mitjançant un espai de cerca (*search space*[49]): el learning rate, el momentum i el weight decay. El *momentum* contribueix a suavitzar les oscil·lacions que caracteritzen l'SGD, facilitant un descens més estable cap al mínim global [67]. El *weight decay*, per la seva banda, actua com un mecanisme de regularització que penalitza pesos excessivament grans, evitant l'overfitting i simulant una reducció progressiva del *learning rate* [67].

Altres hiperparàmetres, com el batch size i el nombre d'epochs, es van mantenir fixos. El *batch size*[50] es va establir en 16, límit màxim que permetia la GPU sense comprometre la memòria. Pel que fa al nombre d'*epochs*[51], es va definir un valor suficientment elevat per **garantir una convergència completa** (al voltant de 60 epochs), tenint en compte que l'entorn d'Ultralytics integra un mecanisme d'*early stopping* que atura automàticament l'entrenament quan ja no s'observen millores significatives en el conjunt de validació [69].

Es van executat 20 iteracions de *tuning* que van tardar aproximadament entre 40 i 48 hores en completar-se.

En resum, el procés d'entrenament s'ha dissenyat per garantir la màxima eficiència i robustesa, combinant tècniques avançades i una planificació acurada de la divisió de dades i la selecció d'hiperparàmetres.

5.2.6 Mètriques d'Avaluació

L'avaluació d'un model de detecció d'objectes no es pot basar en una sola mètrica, ja que cadascuna captura aspectes diferents del seu rendiment, des de la capacitat del model per detectar correctament els objectes com la precisió en la seva classificació i localització. Per aquest motiu, en el projecte s'han emprat diverses mètriques, tant durant la fase de *hyperparameter tuning* com en l'entrenament definitiu [70].

5.2.6.1 Mètriques Bàsiques

Definicions prèvies:

- TP (True Positive): El model prediu un objecte i realment hi és.
- FP (False Positive): El model prediu un objecte però en realitat no hi és.
- FN (False Negative): El model no detecta un objecte que realment hi és.

[49] Search space: conjunt de valors possibles per a un conjunt de variables o hiperparàmetres que es poden explorar durant un procés d'optimització, com el tuning de models, amb l'objectiu de trobar la combinació que maximitza el rendiment.

[50] Batch size: nombre d'exemples processats simultàniament durant una iteració d'entrenament d'un model d'aprenentatge automàtic, que afecta tant l'eficiència computacional com l'estabilitat del procés d'optimització.

[51] Epochs: nombre de vegades que tot el conjunt de dades d'entrenament s'utilitza completament per actualitzar els pesos d'un model durant el procés d'aprenentatge.

- **Precision**: És la proporció de prediccions correctes respecte al total de prediccions realitzades pel model:

$$(4)\ Precision = \frac{TP}{TP+FP}$$

  Si la precisió és alta, significa que quan el model detecta que hi ha un objecte, normalment té raó.

- **Recall**: És la proporció d'objectes reals que el model aconsegueix detectar:

$$(5)\ Recall = \frac{TP}{TP+FN}$$

  Si el recall és alt, el model gairebé no deixa cap objecte sense detectar.

- **F1 Score**: Combina Precision i Recall en una sola mètrica.

$$(6)\ F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

  És útil quan volem un únic valor que reflecteixi tant la capacitat de detectar objectes com la de fer-ho correctament.

- **mAP**: És la mitjana de la Average Precision sobre totes les classes i llindars d'IoU. mAP@50 es calcula amb IoU ≥ 0.5 mentre que mAP@50-95 es calcula amb diferents llindars de 0.5 a 0.95 i després es fa la mitjana.

  És la mesura global del rendiment del model sobre totes les classes. Un mAP alt significa que el model detecta bé i classifica correctament els objectes amb caixes ben ajustades.

5.2.6.2 Fase d'*Hyperparameter Tuning*

Durant el procés de *hyperparameter tuning*, s'han emprat algunes de les mètriques que Ray Tune proporciona per monitorar el rendiment del model:

1. **mAP@50-95** i **mAP@50**
2. **Precision**
3. **Recall.**

Tot i disposar de diverses mètriques, la mètrica principal seguida en aquesta fase va ser mAP@50-95, ja que proporciona una **visió més realista i exigent del comportament del model**.

5.2.6.3 Fase d'Entrenament

En la fase d'entrenament, Ultralytics ofereix informes i visualitzacions que permeten una anàlisi més completa. En aquest cas, s'han considerat les següents mètriques i representacions:

- **Matriu de confusió** (normal i normalitzada): Taula que mostra quantes prediccions per classe han sigut correctes i quantes s'han confós amb altres classes.
- **F1 score** i **F1 curve**
- **Precisió (P)** i **corba P**
- **Recall (R)** i **corba R**
- **Corba PR (Precision-Recall Curve)**: Mostra com canvia la precisió i el recall quan canviem el llindar de confiança del model. El llindar de confiança és el valor mínim que assignem per considerar una predicció com a "vàlida".
  Ens ajuda a trobar l'equilibri òptim entre no perdre objectes (R) i no generar falsos positius (P).
  Una corba més alta indica millor rendiment global: el model manté alta precisió fins i tot amb un recall alt.
- **Box Loss**: Quantifica l'error en la posició i mida de les caixes predictores respecte les reals.
- **Class Loss**: Quantifica l'error en la classificació de les instàncies detectades.
- **Distribution Focal Loss**: Pèrdua que penalitza més els errors en exemples difícils, de manera que el model presta més atenció a objectes poc representats o complicats de classificar, millorant la detecció de classes minoritàries.
- **mAP@50-95** i **mAP@50**

Aquesta combinació de mètriques permet no només quantificar el rendiment global, sinó també identificar els punts febles del model, entendre millor els errors de classificació i avaluar l'impacte del desequilibri entre classes. Les corbes PR i la Distribution Focal Loss són exemples d'eines útils per minimitzar l'efecte del desequilibri en el rendiment del model.

5.2.7 Eines Complementaries

Durant el procés d'entrenament i validació del Model Classificador de Residus, s'han desenvolupat i utilitzat dues eines complementàries que faciliten la gestió i qualitat de les dades d'entrenament ja que aquestes poden tenir cert grau d'error:

5.2.7.1 Labeling Tool

Aquesta eina proporciona una interfície gràfica intuïtiva per a l'etiquetatge manual d'imatges segons el format YOLO:

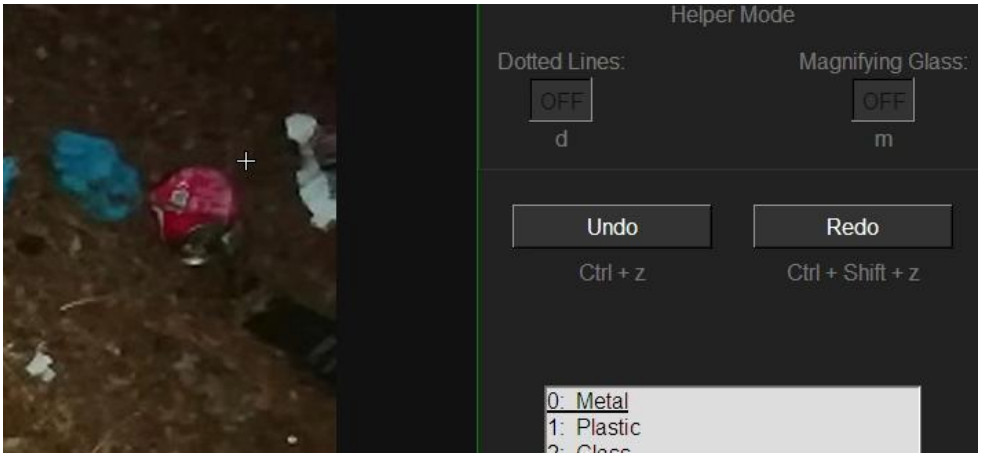
Il·lustració 34. Vista general de la *Labeling Tool*

L'usuari pot seleccionar el directori d'origen de les imatges a anotar i el directori de destí on es guardaran juntament amb les seves anotacions.

És possible dibuixar caixes delimitadores (bounding boxes) amb el ratolí, fent clic dret a un punt inicial i arrossegant el cursor fins delimitar el residu completament. Posteriorment, mitjançant el panell lateral de classes predefinides el sistema permet assignar la categoria corresponent al residu delimitat:

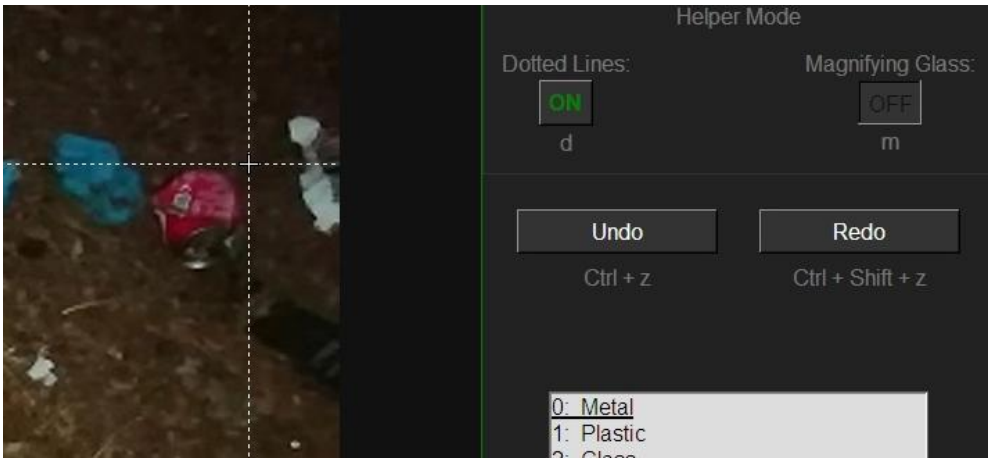
Il·lustració 35. Dibuix de *bounding box* i selecció de residu

Per facilitar un etiquetatge més precís, la Labeling Tool incorpora un **mode d'ajuda**:



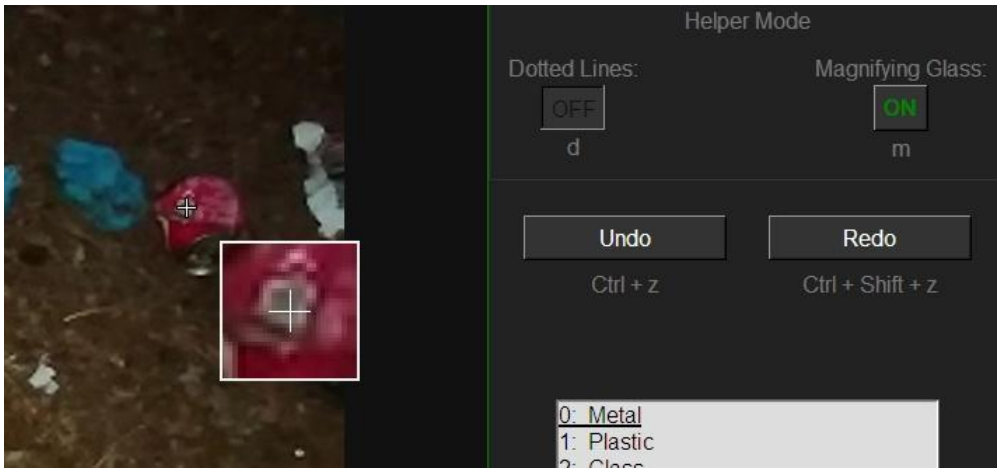
Il·lustració 36. Cursor bàsic sense modes d'ajuda activats

- Línies guia (línies discontínues que s'activen amb la tecla *d*):



Il·lustració 37. Cursor amb línies discontínues activades

- Lupa (s'activa amb la tecla *m*).



Il·lustració 38. Cursor amb lupa activada

Aquests modes d'ajuda es poden utilitzar simultàniament, tot i que no sol ser necessari.

El sistema també permet **gestionar les accions**:

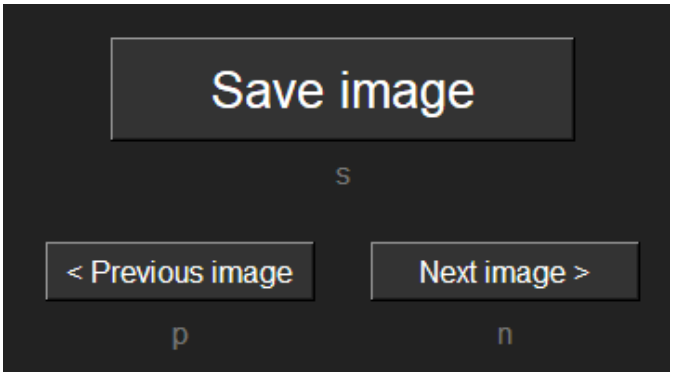
- Desfer i refer (botons *Undo/Redo* o dreceres *Ctrl+Z* i *Ctrl+Shift+Z*).

Il·lustració 39. Botons "Undo" i "Redo"

- Esborrar caixes delimitadores amb el mode goma (*e* o botó corresponent). En cas d'haver-hi caixes superposades, sempre s'elimina primer la més recent.

Il·lustració 40. Botó del mode "goma"

Quan una imatge ja està etiquetada, es pot **guardar** (tecla *s* o botó *Save*). Així mateix, és possible avançar o retrocedir entre imatges sense necessitat de guardar prèviament:



Il·lustració 41. Botons per guardar imatge, i passar a la següent o anterior imatge

Un cop es confirma el guardat, la Labeling Tool desa automàticament les anotacions en el format requerit. Aquestes es mouen juntament amb la imatge a la carpeta destí, preparant-les per a la fase de validació.

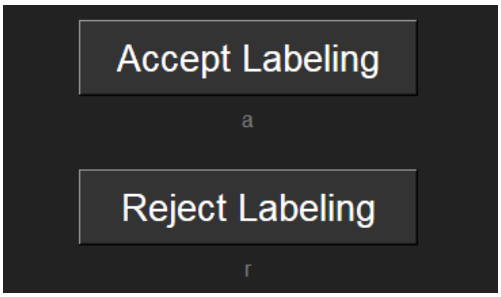
Finalment, està previst integrar aquesta eina dins la pàgina web abocadorsbp.com (quan estigui operativa) amb l'objectiu que els usuaris puguin col·laborar en l'etiquetatge i contribuir així a la millora contínua del model.

5.2.7.2 Label Validation Tool

Un cop les imatges han estat etiquetades, aquesta segona eina permet validar la qualitat de les anotacions mitjançant una interfície gràfica més senzilla que l'anterior:

Il·lustració 42. Vista general de la Label Validation Tool

L'usuari pot revisar cada imatge amb les caixes i categories assignades, i decidir si l'etiquetatge és correcte ("Acceptar", tecla *a*) o si cal rebutjar-lo ("Rebutjar", tecla *r*):



Il·lustració 43. Botons per acceptar o rebutjar anotacions

Les imatges acceptades es mouen a la carpeta definitiva del conjunt d'entrenament, mentre que les rebutjades es traslladen a una carpeta "unlabeled" per a una possible revisió o reetiquetatge. Aquest procés garanteix que només les anotacions de qualitat s'utilitzin en l'entrenament del model.

De manera similar a la *Labeling Tool*, és possible avançar o retrocedir entre imatges amb els botons inferiors del panell de la dreta, sense necessitat d'acceptar o rebutjar prèviament cada anotació.

De cara al futur, aquesta eina tindrà un paper clau dins el flux de treball del portal abocadorsbp.com on tots els etiquetatges realitzats pels usuaris mitjançant la *Labeling Tool* es revisaran de manera centralitzada amb la *Label Validation Tool* per un administrador (l'autor del projecte). Només aquelles anotacions validades passaran a integrar-se al conjunt de dades final. D'aquesta manera, es **manté la coherència i la qualitat del dataset** alhora que es permet la col·laboració oberta de la comunitat.

5.3 Disseny de la Interfície Gràfica

Aquest apartat descriu el disseny i les decisions tecnològiques aplicades a la UI mòbil. El projecte prioritza usabilitat, rendiment i escalabilitat multiplataforma, amb una navegació estructurada per fitxers, components reutilitzables i gestió eficient de dades geoespacials (render map amb WebView + Leaflet i clústers).

5.3.1 Arquitectura de l'Aplicació Mòbil i Decisions Tecnològiques

El desenvolupament de l'aplicació mòbil ha requerit la presa de diverses decisions arquitecturals fonamentals que han condicionat tant l'eficiència del procés de desenvolupament com la qualitat del producte final. Aquestes decisions s'han basat en els criteris establerts a l'inici de l'apartat 5 i l'experiència d'usuari.

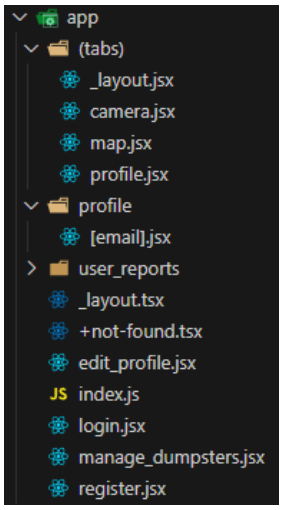
5.3.1.1 Ús d'Expo i Navegació

Per començar, s'ha optat per Expo com a framework de desenvolupament per tres raons principals:

1. Expo proporciona un entorn de desenvolupament preconfigurat que elimina la necessitat de configurar manualment Xcode i Android Studio, accelerant significativament el cicle de desenvolupament.
2. Les actualitzacions OTA[52] permeten desplegar correccions i millores de manera immediata sense passar pels processos d'aprovació de les botigues d'aplicacions, aspecte crucial per a un projecte en fase inicial [71][72].
3. El sistema de *building* automàtic (Expo EAS Build) elimina la complexitat de mantenir configuracions de compilació per a múltiples plataformes [73].

Seguidament, la navegació s'articula mitjançant Expo Router a partir de l'estructura de carpetes dins el directori "app", de manera que cada fitxer es correspon amb una pantalla i la jerarquia reflecteix el flux de l'usuari. Les pestanyes principals es defineixen a (tabs)/, i rutes dinàmiques com profile/[email].jsx i user_reports/[email].jsx faciliten la navegació contextual [74]. Aquesta estratègia fa el codi més llegible i permet escalar l'aplicació afegint simplement nous fitxers a les ubicacions pertinents:

[52] OTA: Over-The-Air



Il·lustració 44. Estructura de fitxers i pantalles de l'aplicació

5.3.1.2 Sistema de Components Reutilitzables

Per evitar el codi repetitiu, l'aplicació depèn d'un sistema de components reutilitzables organitzat en dos nivells: components base i components especialitzats.

Els components base defineixen la capa fonamental de la interfície i **estableixen les regles visuals principals**. Inclouen adaptacions pròpies de components de React Native, com ara *Text* (amb desactivació de l'escalat automàtic i ús de la tipografia Roboto), *ThemedText* (que amplia *Text* amb variants tipogràfiques com *title*, *subtitle* o *link* i integració directa amb el sistema de temes) i *ThemedView* (que aplica automàticament els colors corresponents al tema actiu).

Sobre aquesta base es construeixen els components especialitzats, que **encapsulen funcionalitats concretes** de l'aplicació i es poden reutilitzar en diferents pantalles. A grans trets, aquests components es classifiquen en:

- **Modals funcionals**, orientats a tasques com la gestió d'imatges, la introducció manual d'ubicacions o la classificació de residus.
- **Components de perfil**, centrats en la representació d'estadístiques i informació personalitzada de l'usuari (targetes de dades, gràfics i gestió d'imatges de perfil).

La descripció detallada i l'ús concret d'aquests components dins de la interfície es presenta posteriorment, a l'apartat 5.3.5.

Aquesta estratègia basada en jerarquies de components garanteix consistència visual a totes les pantalles, mantenibilitat millorada, centralitzant canvis estilístics i funcionals, reducció de codi duplicat, gràcies a la reutilització sistemàtica i escalabilitat i testabilitat, en separar clarament la lògica comuna (base) de la funcionalitat concreta (especialitzada).

5.3.1.3 Gestió de Permisos

L'aplicació requereix accés a funcionalitats natives crítiques (càmera, ubicació, emmagatzematge) mitjançant una estratègia defensiva que garanteix funcionalitat fins i tot quan els permisos són denegats.

Per a la càmera (essencial per la classificació de residus) i ubicació, s'implementa sol·licitud progressiva amb explicacions clares de la necessitat del permís. En cas de denegació, l'aplicació ofereix alternatives com la selecció d'imatges de galeria com a alternativa a la càmera i l'ús d'una ubicació predeterminada o la capacitat d'indicar una ubicació exacta a un petit mapa d'ajuda com a alternativa a la ubicació real de l'usuari

La integració amb aquestes APIs natives es realitza mitjançant les llibreries optimitzades d'Expo [36], proporcionant una interfície consistent entre iOS i Android mentre es manté compatibilitat amb les especificitats de cada plataforma.

S'espera que aquesta gestió robusta de permisos optimitzi les taxes de retenció d'usuaris al mantenir l'aplicació completament funcional fins i tot en escenaris restrictius, evitant l'abandonament per part dels usuaris que no vulguin concedir permisos amplis.

5.3.2 Sistema de Temes i Paleta Cromàtica

El disseny visual de l'aplicació es fonamenta en un sistema de temes dual que s'adapta automàticament a les preferències de l'usuari i garanteix una experiència consistent en diferents dispositius i mides de pantalla. Aquest sistema de temes suporta modes clar i fosc.

Es disposa d'un ThemeContext que detecta el tema del sistema en l'arrencada i permet canvis manuals persistits. Aquesta aproximació millora significativament l'experiència d'usuari en diferents condicions d'il·luminació ambiental i respecta les preferències d'accessibilitat individuals, especialment rellevant per a usuaris amb sensibilitat a la llum o que utilitzen l'aplicació en entorns amb poca llum.

El verd corporatiu rgb(0, 150, 0) actua com a tint identitari en ambdós modes, reforçant la temàtica mediambiental i la coherència de marca. La paleta estableix colors per a text, fons, detalls, contorns i fons de camps, maximitzant el contrast i la llegibilitat. Això permet que els components heretin els tons adequats sense configuracions individuals a cada pantalla.

Seguidament es mostren les paletes per ambdós temes, on l'ordre dels colors correspon als camps següents: *tint*, *text*, *background*, *details*, *details2*, *fieldBackground:*

Il·lustració 45. Paleta de colors per al tema clar (Light) de la App

Il·lustració 46. Paleta de colors per al tema fosc (Dark) de la App

5.3.3 Disseny Responsiu

Per a fer l'aplicació responsiva s'ha utilitzat la llibreria *react-native-size-matters*, que resol els problemes d'adaptació a diferents densitats de pantalla i mides de dispositiu. Les funcions *horizontalScale*, *verticalScale* i *moderateScale* calculen automàticament les dimensions adequades basant-se en les dimensions de referència del dispositiu [75].

Aquest sistema és especialment necessari per garantir que l'aplicació mantingui les proporcions visuals correctes tant en dispositius compactes com en *tablets*, evitant que els elements de la interfície apareguin desproporcionats. D'altra banda, la desactivació de l'escalat automàtic de fonts del sistema (gestionat pel component base *Text* esmentat a l'apartat 5.3.1.2) assegura que la tipografia mantingui la seva mida dissenyada independentment de les configuracions d'accessibilitat de l'usuari, preservant la coherència del disseny

Una de les avantatges clau d'utilitzar Expo és la gestió automàtica de la majoria de diferències entre iOS i Android, permetent que l'aplicació funcioni de manera nativa en ambdues plataformes sense configuració manual extensiva. Expo s'encarrega automàticament de l'adaptació de components, gestió de permisos, configuracions de build i optimitzacions específiques de cada plataforma, reduint significativament la complexitat del desenvolupament multiplataforma.

L'aplicació també implementa gestió automàtica de zones segures per adaptar-se a la diversitat de formats de pantalla actuals, des de dispositius amb barres de navegació físiques fins a models amb *notch*[53], *Dynamic Island* [54]o navegació gestual. Aquesta adaptació assegura que tots els elements interactius siguin accessibles i que el contingut es presenti correctament independentment del model específic de dispositiu, eliminant problemes comuns com botons ocults darrere barres del sistema o contingut tallat.

Aquesta estratègia permet concentrar els esforços de desenvolupament en la funcionalitat específica de l'aplicació mentre que Expo gestiona la complexitat inherent del

[53] Notch: Zona retallada de la pantalla d'un dispositiu mòbil destinada a allotjar components físics (com càmera o sensors), que redueix parcialment l'àrea útil de visualització.

[54] Dynamic Island: Element d'interfície introduït en alguns dispositius mòbils que aprofita l'espai del notch mitjançant una àrea dinàmica i interactiva per mostrar notificacions, controls o informació contextual.

desplegament multiplataforma, resultant en una experiència d'usuari polida i nativa en ambdues plataformes.

5.3.4 Gestió d'Estat Global i Optimitzacions

L'aplicació utilitza una arquitectura híbrida que combina dades geoespacials locals amb informació dinàmica del servidor, gestionada mitjançant dos contexts globals especialitzats.

El primer context global, *AsyncGeoDataContext*, gestiona la càrrega asíncrona de fitxers GeoJSON locals que contenen les geometries completes de totes les comarques i municipis de Catalunya [76][77]. Inicialment amb la intenció de tenir un fitxer per regió, però finalment adaptant-se a un fitxer per totes les comarques i un altre per tots els municipis. Aquestes geometries es fan servir en diferents contexts de l'aplicació com per exemple per mostrar-les al mapa delimitant una comarca o municipi o per a calcular la comarca i municipi en la que es troba l'usuari al registrar un abocador.

La decisió d'incloure aquestes dades localment garanteix funcionalitat offline per a informació geoespacial bàsica, elimina latència de xarxa i redueix dependències de serveis externs, tot i que incrementa la mida de l'aplicació.

El segon context, *ProcessedGeoDataContext*, utilitza les dades de comarques i municipis proporcionades per *AsyncGeoDataContext* per crear estructures de dades específiques per a la interfície d'usuari i integrar informació dinàmica del servidor. Aquest context extreu els noms de totes les comarques i municipis dels fitxers GeoJSON locals i realitza peticions POST al servidor per obtenir els recomptes actualitzats d'abocadors reportats en cada àrea geogràfica.

El context genera tres estructures principals: Una llista de comarques ordenades alfabèticament, cada una amb el seu recompte d'abocadors (*comarquesOnly*), una llista equivalent per a municipis (*municipisOnly*), i una llista de comarques com la primera mencionada que afegeix una opció especial "Totes les comarques" amb el recompte total agregat de tots els abocadors (*comarquesDropdown*). Aquestes estructures combinen les geometries GeoJSON locals amb les estadístiques dinàmiques del servidor, proporcionant components de selecció llestos per utilitzar en *dropdowns* i filtres d'abocadors com es veurà a l'apartat 5.3.5.

Aquest disseny híbrid optimitza l'experiència d'usuari proporcionant resposta immediata per a navegació geogràfica i informació actualitzada per a la presa de decisions

5.3.4.1 Arquitectura d'Optimitzacions

Per garantir un rendiment òptim en dispositius mòbils, l'aplicació utilitza estratègies de memoització que permeten evitar càlculs repetits i renders innecessaris minimitzant el consum de recursos. Les dades que requereixen processament intensiu, com les llistes de comarques i municipis amb els seus recomptes d'abocadors, només es recalculen quan realment canvien les dades base. Això assegura que la interfície sigui àgil fins i tot quan es treballa amb grans volums d'informació geoespacial.

A nivell de presentació, la UI memoïtza tant estils (condicionats pel tema i les safe areas) com handlers d'interacció, de manera que els canvis visuals o de filtre no impliquen recomputacions globals.

La memoització s'aplica estratègicament en diversos contexts específics del projecte. Les imatges temàtiques, com les fotos de perfil per defecte, s'optimitzen segons el tema actiu (clar/fosc) per evitar recàrregues innecessàries cada vegada que es navega entre pantalles.

Finalment, les funcions que gestionen interaccions crítiques de l'usuari, com l'obertura de modals, selecció d'imatges, o canvis en dropdowns, s'optimitzen per evitar re-renderitzacions innecessàries dels components fills. Aquesta estratègia és especialment crítica en el projecte perquè combina interfícies complexes (mapes interactius), dades pesades (GeoJSON), i components dinàmics (dropdowns, temes), on qualsevol recàlcul innecessari podria impactar negativament l'experiència d'usuari en dispositius amb recursos limitats.

En resum, la memoització extensiva és clau per garantir que l'aplicació sigui ràpida, eficient i escalable, independentment de la quantitat de dades o la complexitat de la interfície.

Les imatges es tracten de manera específica pel seu impacte en xarxa i memòria. Abans de la pujada es comprimeixen al dispositiu amb *presets*[55] adients (per exemple, foto de perfil), ajustant resolució i qualitat per equilibrar nitidesa, mida final i temps de xarxa.

Amb aquest conjunt d'estratègies (càrrega diferida, agregacions al servidor, memoització de dades i estils, clustering i debounced updates al mapa, i compressió local d'imatges) l'aplicació manté la fluïdesa i conté el consum de recursos, oferint temps de resposta consistents i escalabilitat a mesura que creix el volum d'informació.

5.3.5 Disseny de Pantalles i Experiència d'Usuari

En aquest apartat es descriu el disseny de les pantalles principals accessibles des de la barra de navegació inferior de l'aplicació) i totes les seves pantalles derivades i les decisions preses per optimitzar l'experiència d'usuari. L'objectiu és garantir una navegació intuïtiva i clara, i acomodar a l'usuari de manera que l'enregistrament de nous abocadors sigui un procés amb responsabilitats mínimes i amb recompenses psicològiques per aquest.

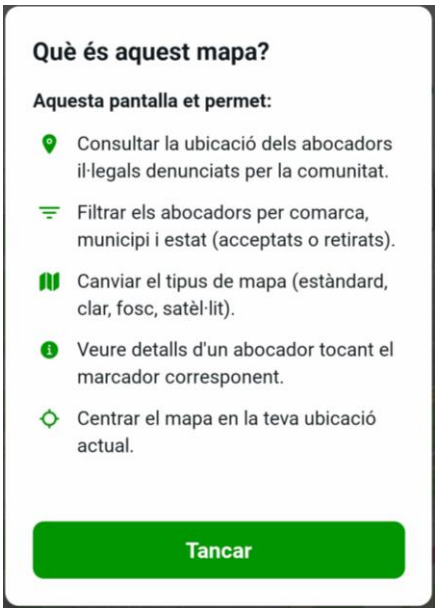
5.3.5.1 Pantalla Inicial – Mapa

La pantalla de Mapa és el punt central de l'aplicació. Aquesta presenta els abocadors en el territori, permet filtrar-los per estat i àmbit geogràfic, i ofereix accés ràpid al detall i a accions administratives. El disseny prioritza la claredat, el *feedback* immediat i la reducció de fricció en les tasques freqüents.

[55] Preset: Configuracions predefinides que estableixen valors o paràmetres per casos específics.

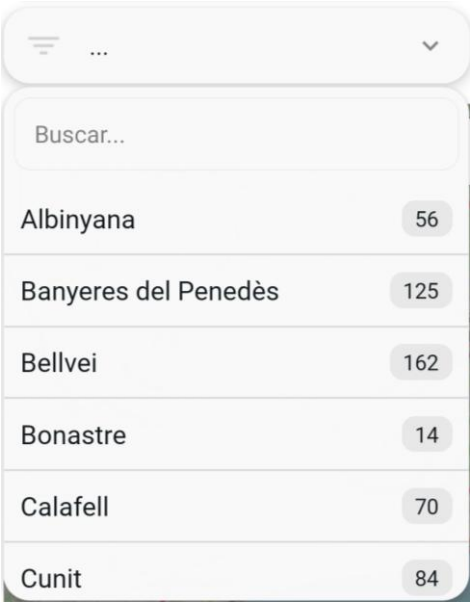
Composició de la interfície:

1. Capçalera minimalista amb títol i botó d'ajuda a la dreta que obre la "Llegenda", un modal breu que explica què es pot fer a la pantalla:



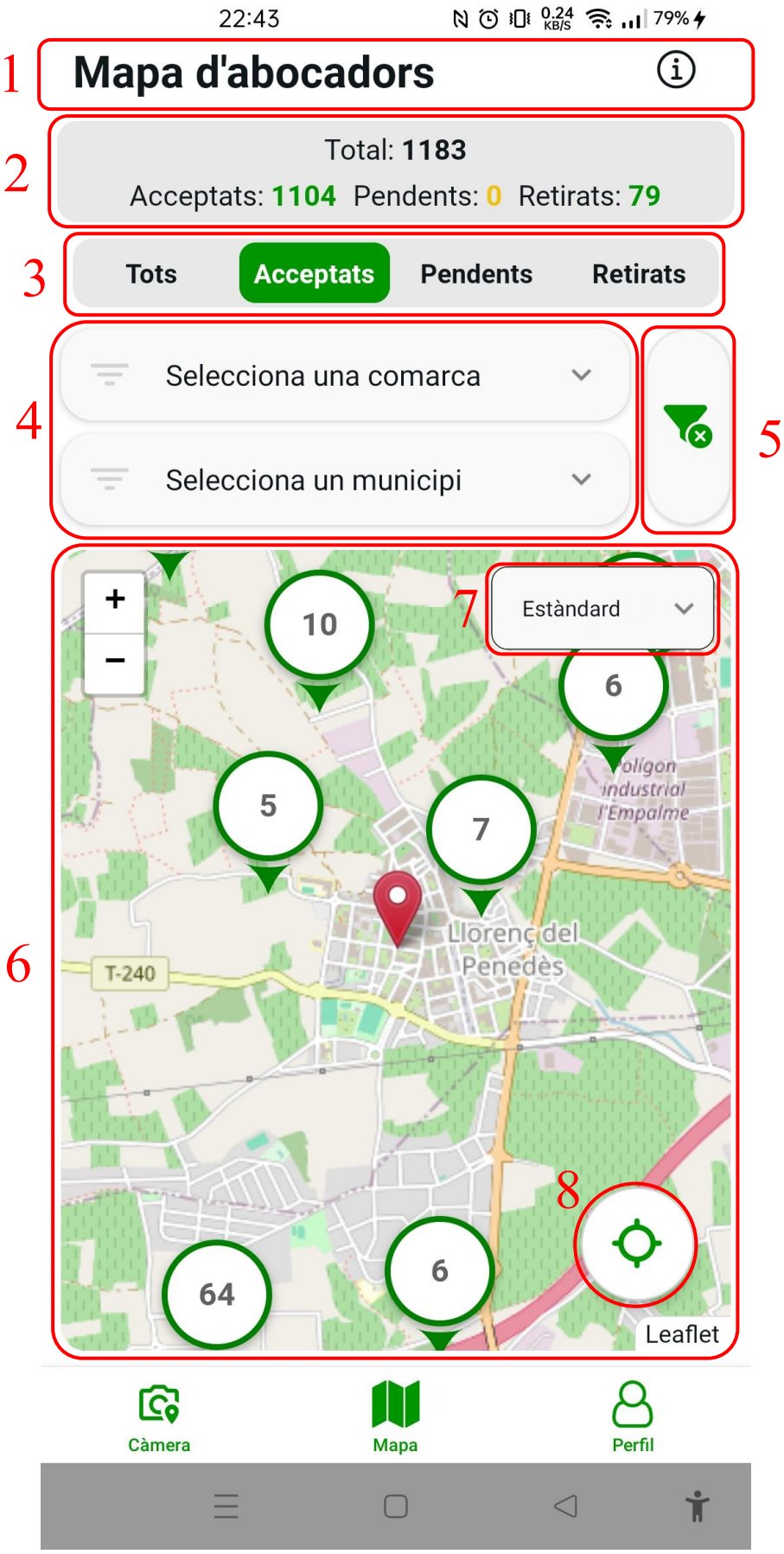
Il·lustració 47. Llegenda d'informació de la pantalla del mapa

2. Barra d'estadístiques ràpides (Total, Acceptats, Pendents, Retirats), que es recalcula segons area seleccionada.
3. Pestanyes de filtre d'estat: Tots, Acceptats, Pendents (visible per administradors) i Retirats.
4. Desplegables per filtrar segons comarca i municipi amb cerca i recompte d'abocadors per element:



Il·lustració 48. Desplegable de municipis amb recompte d'abocadors

5. Botó per netejar selecció de filtres de comarca i municipi i eliminar polígons del mapa.
6. Contenidor del mapa
7. Selector de tipus de mapa (estàndard/clar/fosc/satèl·lit).
8. Botó flotant per centrar el mapa en la ubicació actual



Il·lustració 49. Pantalla del mapa amb elements numerats segons el llistat de composició.

Quan l'usuari inicia l'aplicació, és rebut amb la pantalla principal, centrada en la visualització i gestió dels abocadors mitjançant un mapa interactiu. Inicialment es va considerar la integració amb el servei de **Google Maps**, però aquesta opció requeria l'ús d'una clau d'API (API Key). Tot i que Google ofereix un crèdit mensual gratuït de fins a 200 €, l'excedent d'aquest límit implica costos addicionals o bé la interrupció del servei, fet que podia comprometre la disponibilitat i la sostenibilitat del projecte a llarg termini.

Per aquest motiu, es va optar per reduir la dependència de serveis externs i prioritzar tecnologies gratuïtes i obertes. Concretament, es va escollir **Leaflet**, una llibreria de codi obert en JavaScript àmpliament utilitzada per a la representació de mapes interactius. La seva integració en l'aplicació mòbil es va realitzar mitjançant un **component WebView**, que permet renderitzar el mapa dins de l'entorn de React Native. Les capes cartogràfiques són proporcionades per **OpenStreetMap**, un projecte col·laboratiu que ofereix dades geogràfiques lliures [78]. L'ús d'aquestes capes garanteix independència respecte a proveïdors comercials i permet oferir diferents estils visuals (clar, fosc, estàndard i satèl·lit).

A la cantonada inferior dreta del mapa es mostra l'atribució a **Leaflet**, tal com estableix la llicència d'ús d'aquesta tecnologia, assegurant així el compliment dels requisits legals i ètics associats al programari lliure.

Pel que fa a l'arquitectura de renderització, el costat nadiu calcula i fixa els valors derivats que no canvien en temps real com ara l'extracció de coordenades dels fitxers GeoJSON i la transformació dels polígons de comarques i municipis.

La geolocalització utilitza una estratègia multicapa: Primer intenta obtenir la ubicació GPS, si la ubicació no es pot resoldre s'inicia un *fallback*[56] a Barcelona (41.3851, 2.1734). Aquesta aproximació garanteix que els usuaris puguin consultar abocadors independentment de l'estat dels seus permisos d'ubicació.

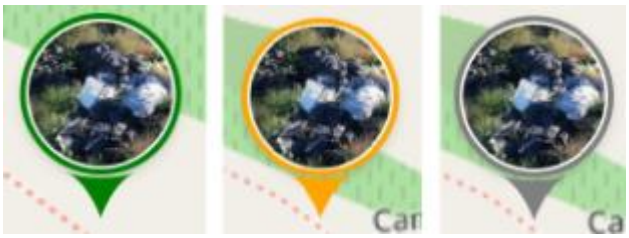
D'altra banda, els esdeveniments de moviment i zoom s'atenen amb un petit *debounce*[57] de 400 mil·lisegons per agrupar canvis i evitar ràfegues de peticions minimitzant el treball de React Native mentre l'usuari es mou pel mapa.

La comunicació entre React Native i la WebView del mapa es gestiona mitjançant un sistema de missatgeria bidireccional que separa clarament les responsabilitats. React Native s'encarrega de la lògica de negoci, l'estat de l'aplicació i la gestió de dades, mentre que Leaflet maneja exclusivament la renderització cartogràfica i la interacció directa amb el mapa. Aquesta divisió permet mantenir el rendiment òptim en ambdós entorns.

[56] Fallback: mecanisme de substitució que s'activa quan una funció, procés o servei principal falla o no està disponible, garantint la continuïtat del funcionament amb una alternativa menys optimitzada o provisional.

[57] Debounce: Tècnica de control d'esdeveniments que limita l'execució d'una funció, ajornant-la fins que ha transcorregut un interval determinat des de la darrera activació.

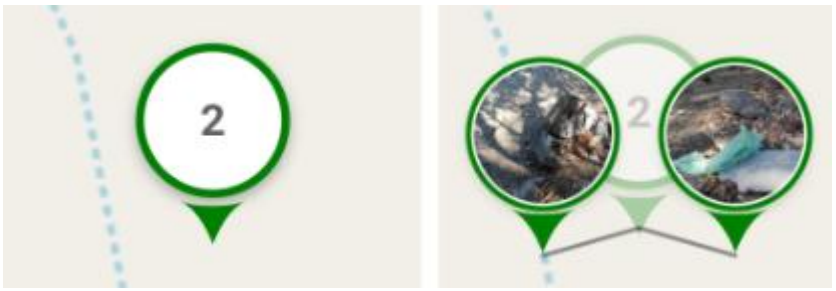
Pel que fa a la representació dels abocadors, cada punt del mapa es mostra amb un marcador visualment expressiu que prioritza la llegibilitat i la identificació ràpida. La miniatura d'imatge actua com a pista visual immediata del contingut, mentre que el codi cromàtic del marcador (verd per a Acceptat, taronja per a Pendent i gris per a Retirat) reforça l'estat de cada denúncia sense obligar l'usuari a llegir detalls:



Il·lustració 50. Previsualització de marcadors al mapa

Quan la imatge encara no és disponible, es mostra un indicador discret d'espera, mantenint la UI coherent i evitant buits visuals. Aquesta decisió redueix la càrrega cognitiva, accelera el reconeixement i aporta una recompensa visual que anima la participació.

Per evitar saturació i millorar l'experiència en dispositius mòbils, els marcadors es combinen automàticament en clústers quan hi ha alta densitat en pantalla. El clúster mostra el recompte agregat i, a mesura que l'usuari s'apropa, es desplega de forma progressiva per facilitar la selecció precisa. Quan diversos punts coincideixen gairebé en la mateixa ubicació (separats a menys de 17 metres aproximadament), la disposició radial (*spiderfy* o espigolació, en català) evita solapaments i permet tocar amb exactitud el punt desitjat. El resultat és un mapa més net, amb millor punteria tàctil i sense pèrdua de context:



Il·lustració 51. Exemple d'espigolació

Això es complementa amb una càrrega incremental i progressiva acotada al *viewport*[58] on es demanen primer les dades de tots els abocadors dins la part visible del mapa i, seguidament, només les imatges dels elements realment visibles (marcadors no agrupats en clústers i marcadors dins de clústers que hauran d'espigolar-se), mostrant un indicador temporal fins que arriben. Aquestes dades i imatges es mantenen en memòria cau per a navegacions posteriors, minimitzant l'ús de dades i temps d'espera. Aquesta estratègia fa el mapa més lleuger i fluid, especialment rellevant en xarxes mòbils i dispositius amb recursos limitats.

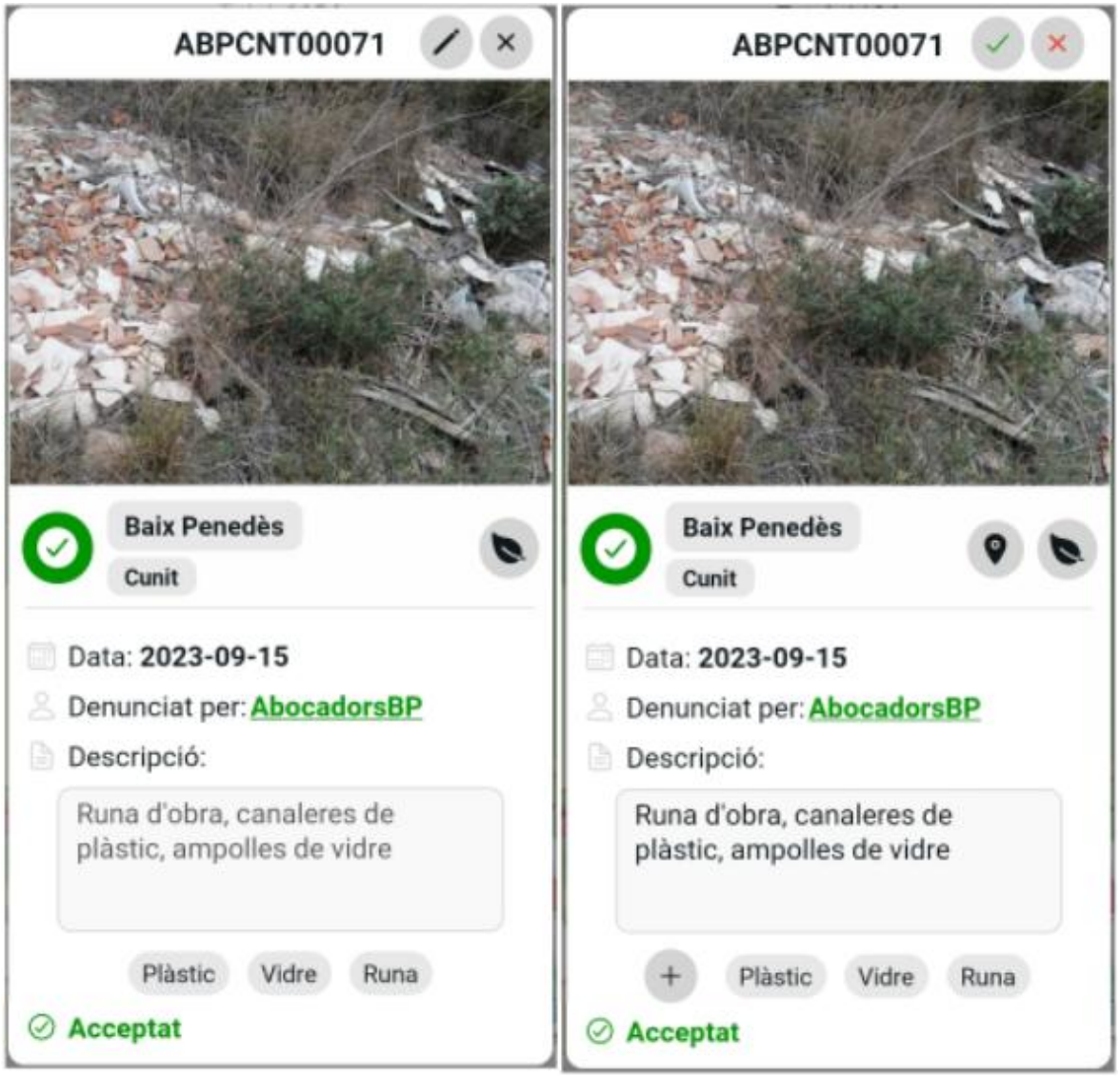
[58] Viewport: Àrea visible d'una interfície gràfica o d'un document digital en què es mostra el contingut

La interacció es manté contextual i no intrusiva: en tocar un marcador, s'obre un panell de detall en modal amb la informació essencial de l'abocador:

1. Nom autogenerat de l'abocador.
2. Botó d'edició (icona de llàpis), només visible per usuaris que siguin administradors.
3. Botó per tancar el modal d'informació.
4. Imatge principal que actua com a referència visual.
5. Informació administrativa que conté la comarca i municipi associats.
6. Botó de retirada (icona de fulla), accessible per a tots els usuaris mentre l'abocador no estigui marcat com a retirat.
7. Data de la denúncia
8. Nom del denunciant, que en clicar-se redirigeix a l'usuari al perfil del denunciant.
9. Descripció textual de l'abocador (pot estar buida ja que és opcional).
10. Residus detectats, automàtica i manualment
11. Estat de l'abocador

Il·lustració 52. Modal d'informació d'abocador amb elements numerats.

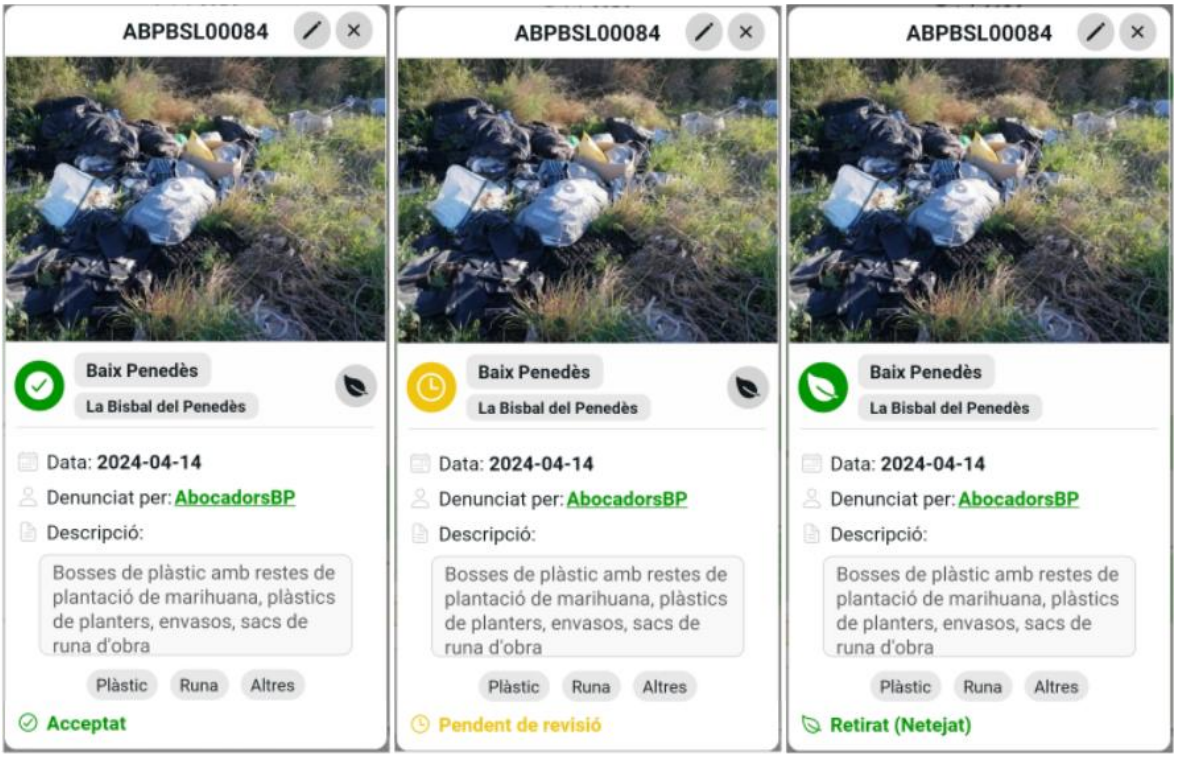
Quan un administrador clica el botó d'edició **(2)**, el modal entra en **mode edició** *in situ*. En aquest estat, la descripció passa de ser un camp només de lectura a un camp editable, i s'activen els controls d'entrada corresponents. Paral·lelament, els botons d'edició i de tancar el modal són substituïts per dos nous botons, un per confirmar les modificacions i un altre per cancel·lar-les i retornar a l'estat de lectura. A més, a l'esquerra del botó de retirada apareix un botó addicional per modificar la ubicació de l'abocador que recomputa automàticament la comarca i municipi quan es modifica la ubicació, mitjançant un algoritme *point-in-polygon*[59] (PIP) sobre les geometries locals. Finalment, al costat esquerre del llistat de residus, el botó amb el símbol "+" permet afegir o eliminar residus de manera senzilla i intuïtiva:



Il·lustració 53. Comparativa entre mode d'edició desactivat i activat respectivament

[59] Point-in-polygon: Algorisme clàssic de computació geomètrica que permet determinar si un punt amb coordenades donades es troba dins o fora d'un polígon.

L'estat s'assenyala amb iconografia i color coherents amb la identitat visual del projecte:



Il·lustració 54. Modal d'informació d'abocador amb diferents estats

El sistema de gestió d'estats implementa una lògica de permisos granular que respecta els rols d'usuari. Mentre que qualsevol usuari registrat pot sol·licitar la retirada d'un abocador (activant un workflow de notificació per correu electrònic a l'equip d'administració), només els administradors poden accedir a funcions de moderació avançada com l'edició directa de metadades o el canvi d'estat immediat. Aquesta separació de responsabilitats garanteix la integritat de les dades alhora que manté la participació ciutadana oberta.

La retroalimentació visual és immediate i contextual ja que cada acció genera confirmacions clares (alerts de sistema) que informen sobre l'èxit o fracàs de l'operació, evitant que l'usuari quedi en estat d'incertesa.

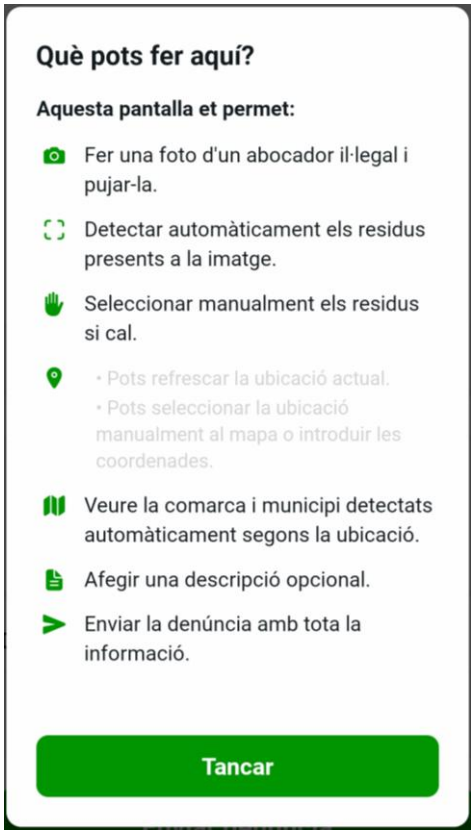
El modal conserva l'estat de navegació de l'usuari. Quan aquest es tanca, el mapa retorna exactament a la mateixa posició i zoom que tenia abans d'obrir-se, preservant el context espacial. Aquesta continuïtat redueix la desorientació i facilita l'exploració iterativa del territori

5.3.5.2 Pantalla d'Enregistrament – Camera

La pantalla de Camera constitueix el nucli operatiu de l'aplicació per a la creació de noves denúncies. El seu disseny prioritza un flux lineal i intuïtiu que guia l'usuari des de la captura d'imatge fins a l'enviament final, minimitzant la fricció i maximitzant la taxa de completació. L'objectiu és transformar el procés de denúncia en una experiència àgil per reduir les responsabilitats de l'usuari sobre el registre.

Composició de la interfície:

1. Capçalera minimalista amb títol i botó d'ajuda a la dreta que obre la "Llegenda", un modal breu que explica què es pot fer a la pantalla:



Il·lustració 55. Llegenda d'informació de la pantalla de la càmera

2. Secció d'imatge amb previsualització amb relació d'aspecte 4:3 del *placeholder* tematitzat segons el mode (fosc/clar):

Il·lustració 55. *Placeholders* d'imatges al registre de denúncies

3. Botó de detecció automàtica de residus amb el model YOLO, superposat a la previsualització.
4. Contenidor de residus detectats amb llistat dinàmic o estat buit "Cap residu a mostrar".
5. Botó d'addició manual ("+") per a selecció complementaria de residus.
6. Informació d'ubicació amb coordenades, comarca i municipi detectats automàticament.
7. Botons de gestió d'ubicació: refresh per reobrir GPS i location per ajust manual
8. Camp de descripció multilínia opcional.
9. Botó d'enviament de denúncia.

Il·lustració 56. Pantalla de la càmera amb elements numerats segons el llistat de composició.

Abans d'accedir a les funcionalitats de la pantalla, el sistema verifica automàticament que l'usuari estigui autenticat mitjançant la presència d'un email vàlid a l'emmagatzematge local (*AsyncStorage*). En cas que no es detecti una sessió activa, l'usuari és redirigit automàticament a la pantalla de login, garantint que només usuaris registrats puguin crear noves denúncies. Això és degut a que el registre d'abocament necessita un ID d'usuari per tal de poder relacionar un abocador amb el seu denunciant.

Quan l'usuari selecciona una imatge, ja sigui des de la càmera o la galeria mitjançant el component *ImagePickerModal* (es veurà més endavant a l'apartat **¡Error! No se encuentra el origen de la referencia.**), aquesta es retalla per a tenir relació d'aspecte 4:3 i assegurar consistència al llarg de totes les imatges assegurant així que no hi hagi errors visuals o artefactes a les diferents vistes de la app. Després s'inicia un procés de compressió intel·ligent al dispositiu. Durant la compressió, es mostra un *overlay*[60] visual amb indicador de progrés i el text "Comprimint...", proporcionant feedback immediat a l'usuari. Si el procés de compressió falla per qualsevol motiu tècnic, el sistema implementa una estratègia de *fallback* que utilitza la imatge original sense comprimir, garantint que l'usuari pugui continuar amb la denúncia sense interrupcions al cost d'enviar i guardar més dades.

En referència a l'ús del model de detecció de residus inicialment es va pensar que estaria integrat localment al dispositiu mòbil però ràpidament es va veure que no era una opció viable, Per tant, Quan l'usuari prem el botó "Detectar residus", la imatge comprimida s'envia al backend per a la seva predicció. La resposta del model es gestiona de forma robusta: en cas d'èxit, es mostren els residus detectats; en cas d'error o absència de deteccions, es presenta un estat buit amb el missatge "Cap residu a mostrar", mantenint sempre una interfície coherent i predictible.

Reconeixent les limitacions inherents de qualsevol sistema de detecció automàtica, s'ha implementat un mecanisme de selecció manual accessible mitjançant el botó "+" posicionat estratègicament al contenidor de residus. Aquest obre el component *ManualWasteModal* (es veurà més endavant a l'apartat **¡Error! No se encuentra el origen de la referencia.**), que permet a l'usuari afegir, eliminar o modificar la selecció de residus de forma intuïtiva.

Aquest enfocament híbrid (automàtic + manual) assegura que l'usuari mantingui sempre el control final sobre la classificació, millorant tant la precisió de les dades com la confiança en el sistema.

La determinació de la ubicació segueix una estratègia multicapa que combina automatització amb control manual. A l'inici de la pantalla, el sistema sol·licita permisos de geolocalització i, un cop obtinguda la posició GPS, executa algoritmes de punt-en-polígon (point-in-polygon) sobre les geometries GeoJSON locals de comarques i municipis per determinar automàticament la jurisdicció administrativa. Cada càlcul geogràfic es

[60] Overlay: element visual superposat sobre la interfície principal que centra l'atenció de l'usuari en una acció sense perdre completament el context de la pantalla subjacent.

complementa amb indicadors de càrrega específics (spinners) que informen l'usuari sobre l'estat del procés: "carregant ubicació", "carregant comarca", "carregant municipi".

Aquest processament es realitza en temps real al dispositiu, de manera que quan l'usuari ja ha seleccionat una imatge per l'abocador i ha definit els residus amb ajuda del model de IA, les dades referents a la ubicació ja estan disponibles.

Si la geolocalització automàtica falla o l'usuari desitja ajustar la posició, el botó de ubicació manual obre el component *ManualLocationModal* (es veurà més endavant a l'apartat **¡Error! No se encuentra el origen de la referencia.**), que integra un selector c artogràfic interactiu i camps d'entrada directa de coordenades. Qualsevol modificació manual desencadena una recomputació automàtica de comarca i municipi, mantenint sempre la coherència de les dades administratives.

El camp de descripció presenta una mida fixa i incorpora un mecanisme de desplaçament intern (*scroll*) per garantir-ne la usabilitat quan el text introduït supera l'espai visible. De manera anàloga, si la llista de residus excedeix l'alçada disponible, la pantalla activa un mode de desplaçament vertical que assegura que tots els elements romanguin accessibles per a l'usuari.

Finalment, abans de permetre l'enviament, el sistema executa una sèrie de validacions exhaustives que garanteixen la integritat de les dades:

- **Presència d'imatge**: verificació que s'ha seleccionat i processat correctament una fotografia
- **Ubicació completa**: validació de coordenades numèriques vàlides i resolució exitosa de comarca i municipi
- **Classificació de residus**: confirmació que s'ha seleccionat almenys un tipus de residu
- **Descripció sanejada**: aplicació de funcions de sanitització i validació de longitud màxima

Les coordenades es validen mitjançant funcions específiques que comproven els rangs geogràfics vàlids, mentre que la descripció textual es processa per assegurar límits de caràcters i evitar injeccions SQL entre d'altres.

En cas de que alguna dada sigui incorrecta, s'envia un missatge d'alerta a l'usuari.

L'enviament final construeix un *payload* JSON[61] complet que inclou la imatge en base64, coordenades geogràfiques, informació administrativa, data en format ISO, descripció sanejada, identificador del denunciant i llista de residus classificats. En cas de que la denúncia hagi sigut registrada per un administrador, el sistema considera que la denúncia és instantàniament vàlida i el servidor la guarda amb estat Acceptat. En cas contrari, la denúncia es marca amb estat Pendent i haurà d'esperar a que un administrador la validi per a aparèixer al mapa per a la resta d'usuaris no administradors.

[61] JSON: JavaScript Object Notation

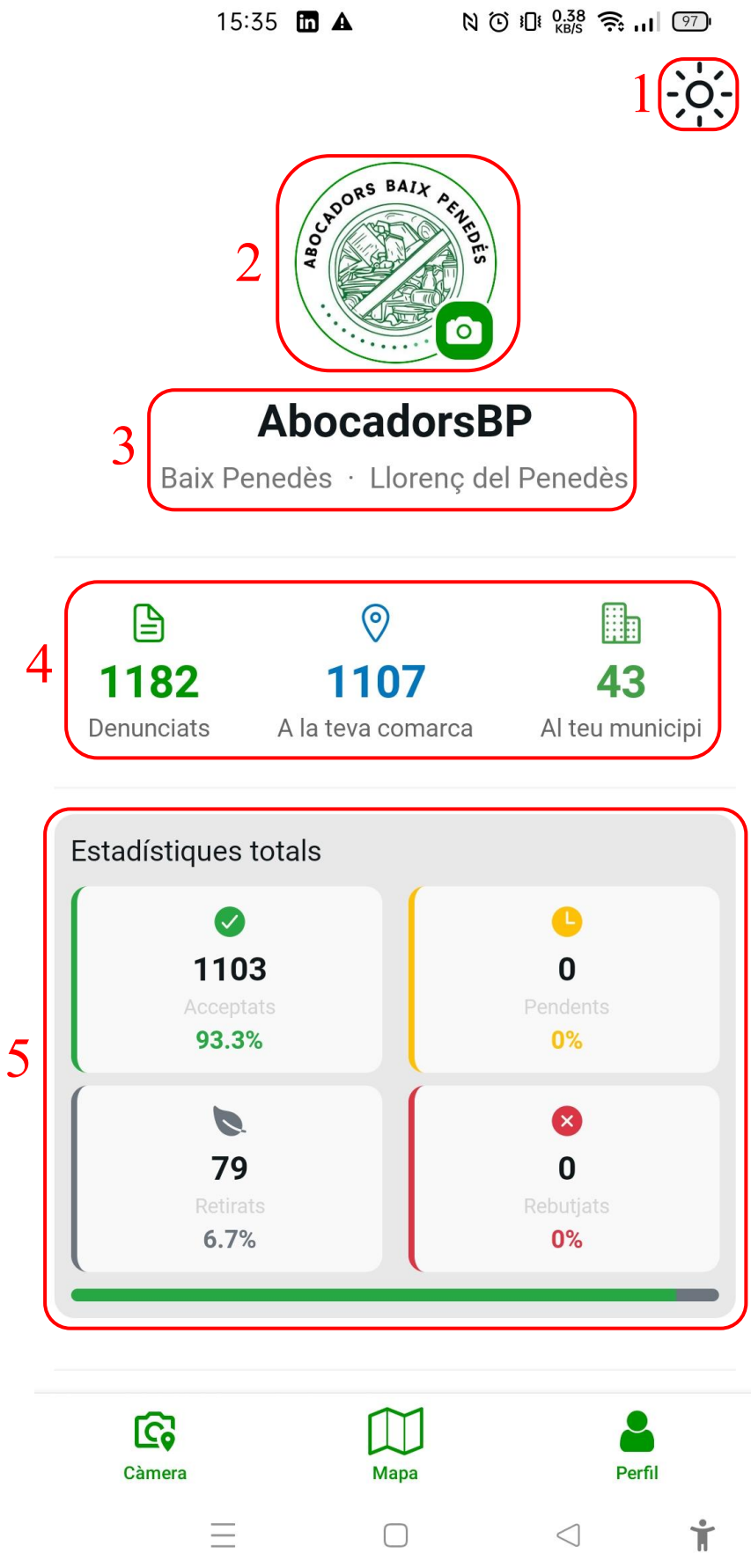
En cas d'èxit, l'usuari rep una confirmació visual mitjançant una alerta i el formulari es reinicia automàticament (neteja d'imatge, residus i descripció), preparant la interfície per a una nova denúncia. D'aquesta manera s'assegura un flux lineal i visible: imatge → residus → ubicació → descripció → enviar.

5.3.5.3 Pantalla de Perfil

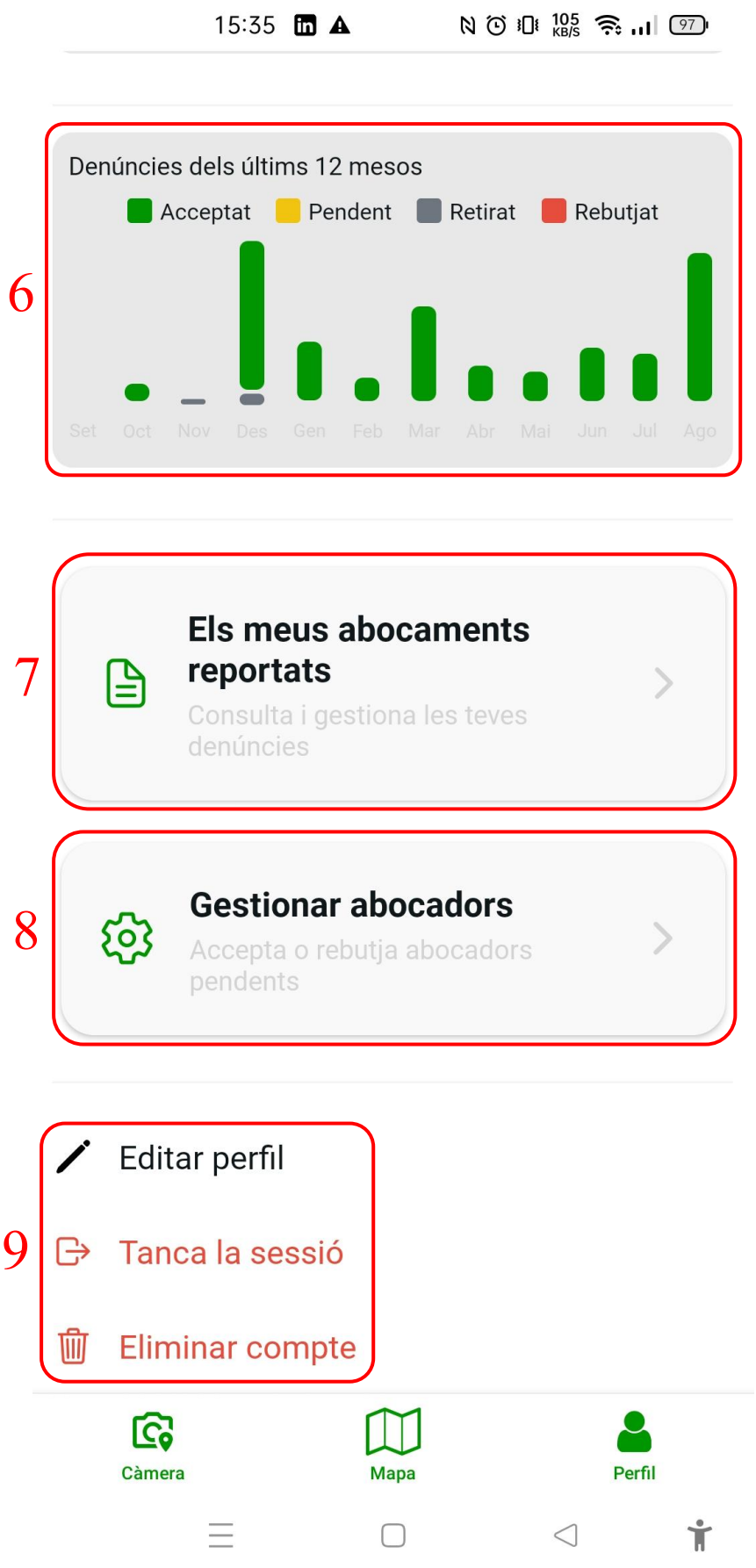
La pantalla de Perfil concentra la personalització de compte, la visualització d'estadístiques i l'accés a accions clau (gestió d'imatge, temes, denúncies, sessió i eliminació de compte). El disseny prioritza la claredat, la persistència d'estat i el feedback immediat.

Composició de la interfície

1. Commutador de tema a la capçalera dreta amb icona temàtica (sol/lluna).
2. Imatge de perfil editable.
3. Informació d'identitat com el nom d'usuari i localització "Comarca · Municipi".
4. Targeta d'estadístiques personals amb totals global, comarcal i municipal de l'usuari.
5. Targeta d'estadístiques detallades de l'usuari amb percentatges.
6. Gràfic de barres mensuals amb sèries per estat (acceptat, pendent, retirat, rebutjat).
7. Targeta d'accés a les denúncies de l'usuari.
8. Botó de gestió d'abocadors (visible per administradors).
9. Opcions de perfil: editar perfil, tancar sessió i eliminar compte.



Il·lustració 57. Part superior de la pantalla del perfil amb elements numerats segons el llistat de composició.

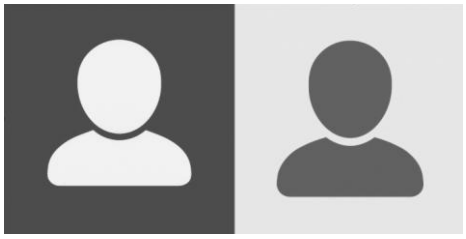


Il·lustració 58. Part inferior de la pantalla del perfil amb elements numerats segons el llistat de composició.

Abans d'accedir a qualsevol funcionalitat del perfil, el sistema verifica automàticament la presència d'un email vàlid a l'emmagatzematge local (AsyncStorage). En cas d'absència de sessió activa, l'usuari és redirigit immediatament a la pantalla de login, garantint que només usuaris autenticats puguin accedir a les dades personals i funcions sensibles.

L'arquitectura de dades implementa una estratègia híbrida que combina emmagatzematge local amb sincronització remota per optimitzar rendiment i experiència d'usuari. Les dades essencials es mantenen en cache a AsyncStorage sota claus descriptives per cada element de la següent llista. En aquesta llista es mostra el valor per defecte que fa servir l'aplicació en cas de que la clau estigui buida:

- **Imatge de perfil**: Per defecte és cadena buida " que la app detecta com a indicador de que ha de posar un *placeholder* tematitzat:



Il·lustració 59. *Placeholders* d'imatges de perfil

- **Nom d'usuari**: "Nom d'usuari no definit".
- **Comarca**: "Comarca no definida".
- **Municipi**: "Municipi no definit".
- **Tema**: Tema predeterminat del sistema.

També es disposa d'una clau que indica si l'usuari és administrador o no, que per defecte es considera falsa. Com s'ha comentat anteriorment, el sistema emmagatzema una clau per al correu que conté l'identificador principal de l'usuari i serveix com a referència per a totes les operacions d'API.

Quan alguna d'aquestes dades està absent o incompleta al cache local, el sistema demana la informació completa del servidor.

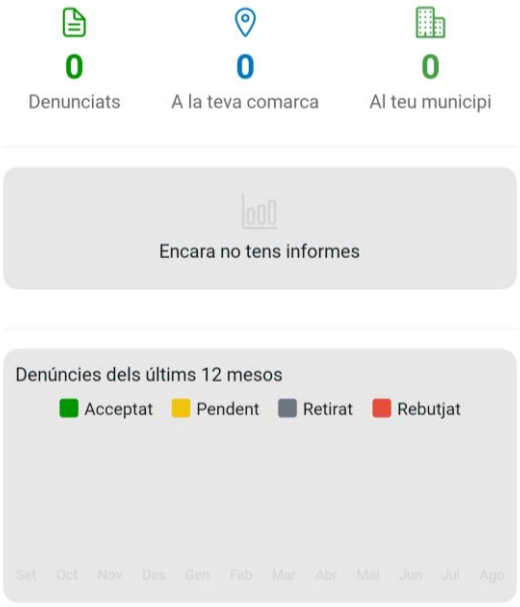
El commutador de tema, posicionat estratègicament a la capçalera dreta, permet alternar entre els modes clar i fosc amb persistència automàtica. Cada canvi s'aplica immediatament a la interfície i es desa a la memòria garantint que la preferència es mantingui entre sessions d'ús. La iconografia del commutador és contextual ja que mostra una icona de sol en mode clar i una icona de lluna en mode fosc, proporcionant una pista visual clara sobre l'estat actual.

Estadístiques i gamificació

La pantalla presenta tres nivells d'informació estadística complementaris: estadístiques personals generals, detallades i evolució temporal mensual. Aquestes estadístiques estan pensades per a *gamificar* l'experiència d'usuari, incentivant així a que la comunitat aporti denúncies reals i precises i evitant denúncies falses, vagues o mal fetes.

Les estadístiques personals generals estan pensades per descriure a grans trets l'ús que fa l'usuari al voltant del seu nucli usual mentre que les detallades aporten certa noció de la qualitat de les denúncies que enregistra ja que mostra el percentatge de denúncies de l'usuari segons estat de denúncia i una barra horitzontal que permet visualitzar aquests percentatges. Finalment, l'evolució temporal mensual mostra un gràfic de barres que permet visualitzar la quantitat de denúncies que l'usuari ha fet en els últims 12 mesos. El gràfic implementa escalat automàtic basat en el valor màxim mensual i utilitza una paleta cromàtica coherent amb la identitat visual del projecte. Aquestes estadístiques d'evolució temporal serveixen per incitar a l'usuari a tenir constància.

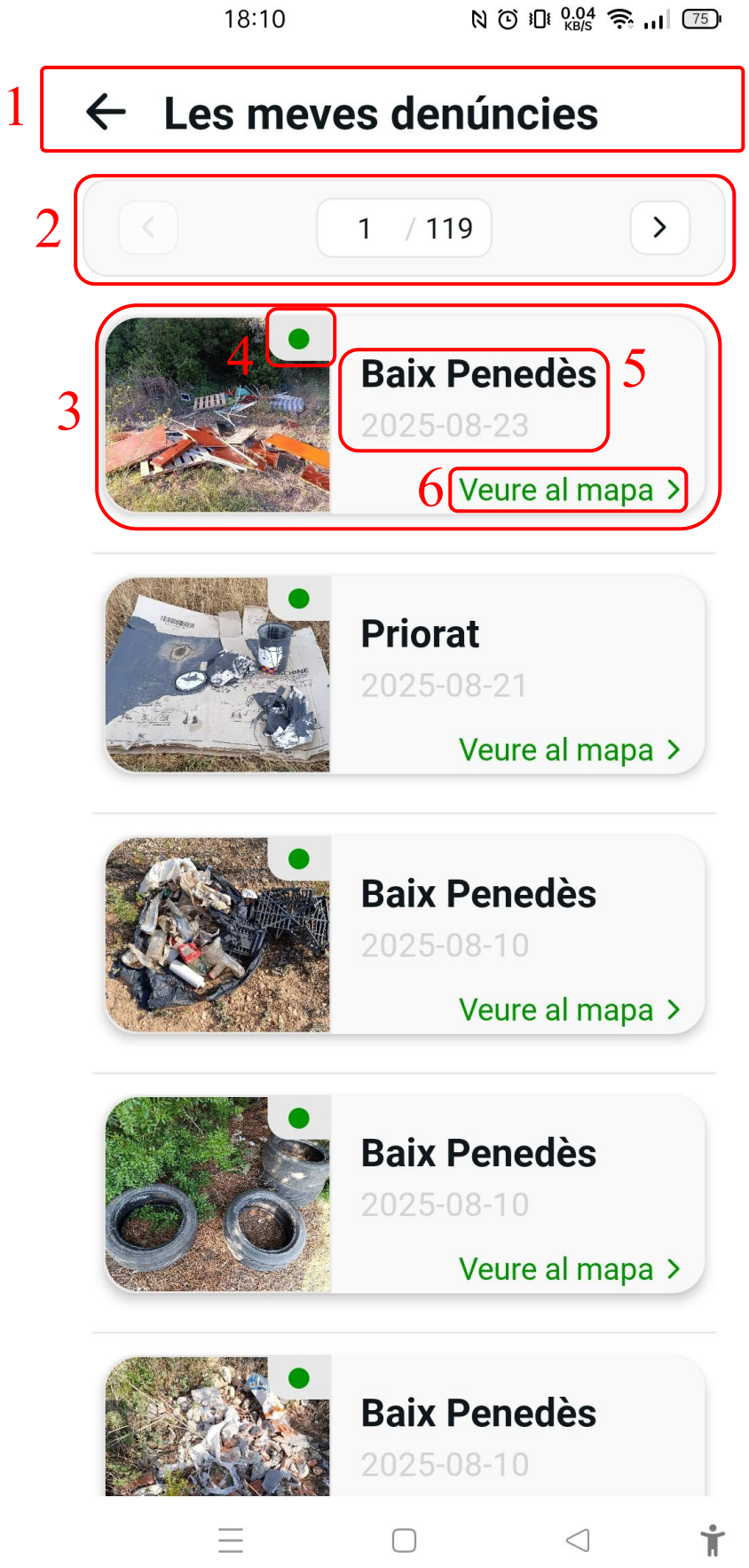
D'altra banda, si no existeixen denúncies, l'aplicació mostrarà els comptadors a 0 en el cas de les estadístiques generals, un missatge informatiu en el cas de les detallades i finalment, una gràfica buida per a l'evolució temporal:



Il·lustració 60. Estadístiques buides

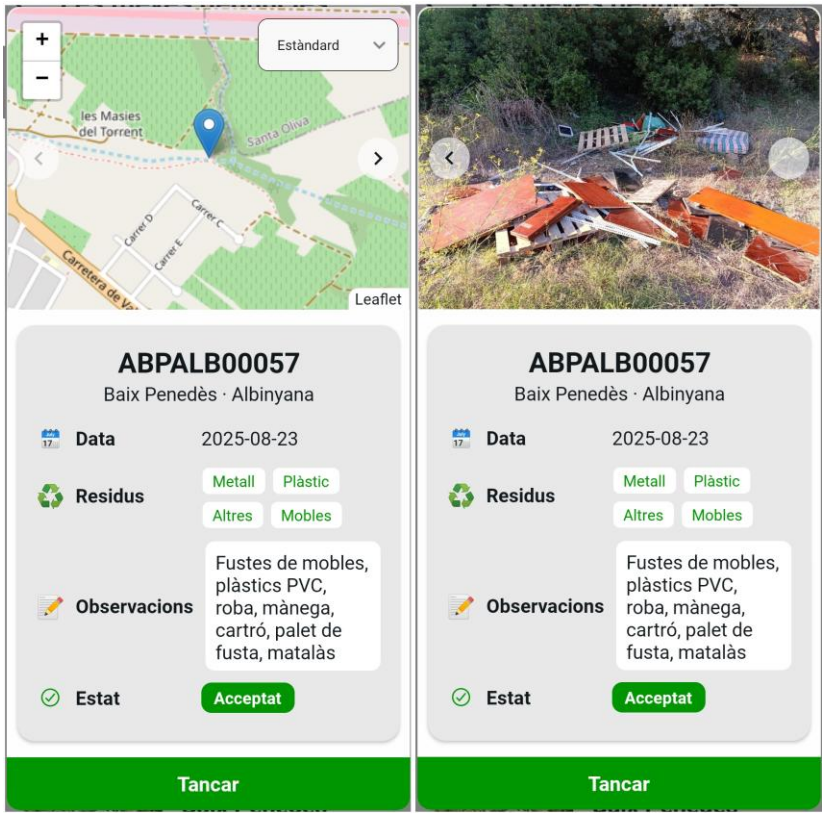
La targeta d'informes personals (7) actua com a pont cap a la previsualització de les denúncies de l'usuari. Aquí trobem una llista paginada on cada pàgina conté 10 elements. D'aquesta manera, quan es carrega una pàgina només es demanen les dades de 10 abocadors, en comptes de les dades de tots els abocadors que l'usuari hagi denunciat. Més concretament en aquesta vista trobem:

1. Capçalera amb fletxa per tornar enrere i títol "Les meves denúncies".
2. Capçalera amb botons per la paginació als costats i selector i indicador de pàgina al centre.
3. Elements previsualitzats.
4. Visualitzador d'estat de la denúncia.
5. Comarca i data de denúncia.
6. Link per obrir un modal amb informació més extensa.



Il·lustració 61. Pantalla de llistat de denúncies d'usuari amb elements numerats.

La pantalla permet a l'usuari clicar l'enllaç "Veure al mapa" (6) per visualitzar amb més detall un abocador de la llista. Aquest enllaç obre un modal que mostra tota la informació relacionada amb l'abocador, així com un contenidor multimèdia que pot alternar entre el mapa interactiu bàsic amb la seva ubicació i la imatge corresponent. El canvi entre vistes es realitza mitjançant els botons *chevron* situats a ambdós costats del contenidor:



Il·lustració 62. Modal de denúncia d'usuari amb vista alternada entre mapa i imatge

Tornant a la pantalla de perfil, per als usuaris amb privilegis administratius, es presenta la targeta "Gestionar abocadors" (8) que proporciona accés directe a les funcions de moderació. La visibilitat d'aquest control es determina mitjançant la clau emmagatzemada localment per determinar si un usuari és administrador, verificada contra les dades del servidor. D'aquesta manera afegim una doble capa de seguretat per a que només els administradors puguin gestionar els abocadors pendents dels usuaris de la app.

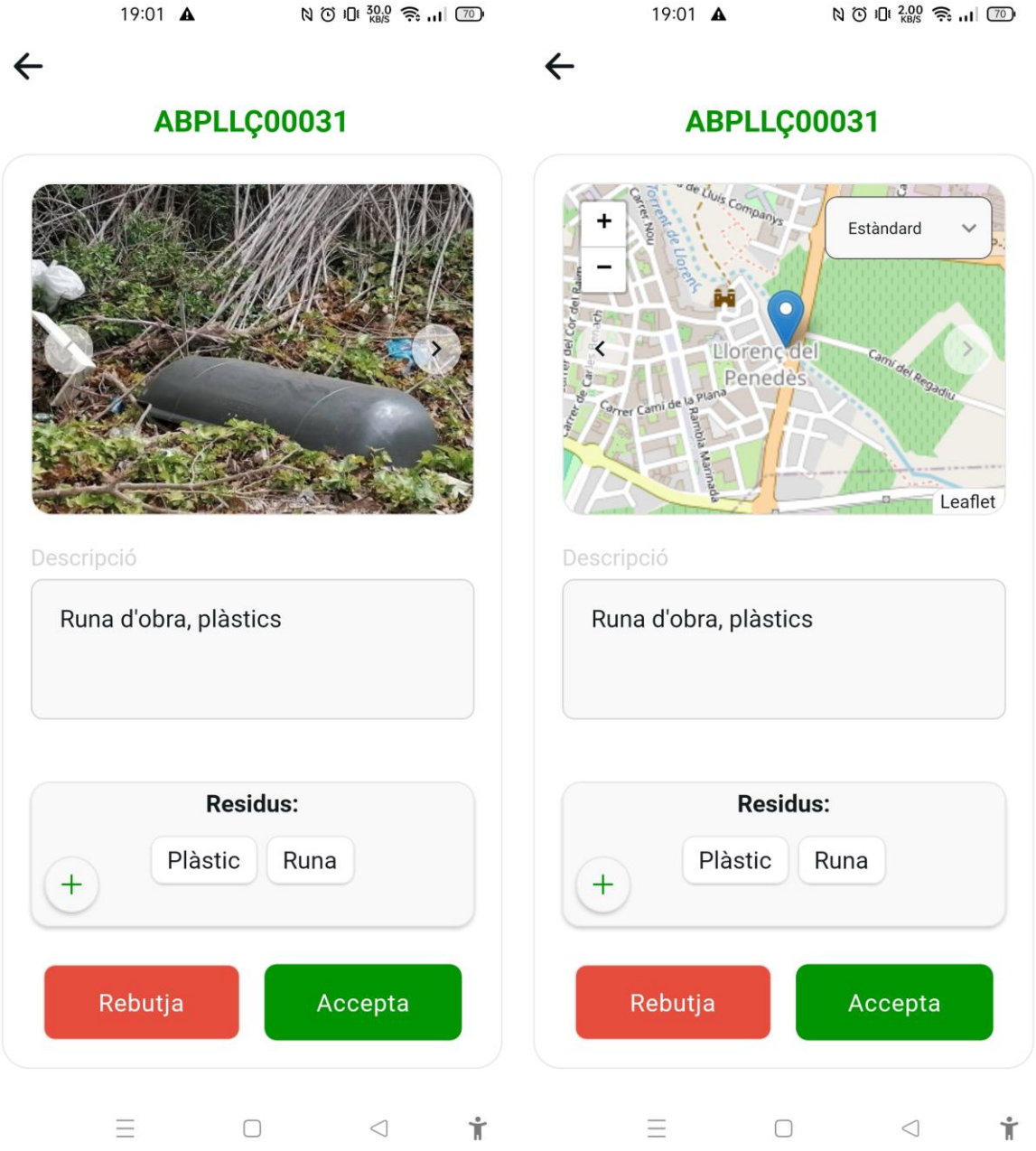
Quan un administrador accedeix a la gestió d'abocadors, el sistema demana al servidor, de forma paginada, els abocadors que estiguin marcats amb estat Pendent. D'aquesta manera s'evita sobrecarregar la memòria del dispositiu i reduir el tràfic de xarxa ja que a la llarga, hi poden arribar a haver milers o desenes de milers d'abocadors pendents, cosa que podria fer col·lapsar el dispositiu. A mesura que l'administrador va gestionant abocadors pendents, la aplicació en va demanant de nous al servidor, creant una experiència fluida per a l'administrador.

L'administrador pot lliscar el dit cap a la dreta o l'esquerre per passar al següent o anterior abocador pendent sense haver d'acceptar-lo o rebutjar-lo.

L'aplicació permet a l'administrador revisar la imatge i la ubicació de l'abocador, per verificar que no es tracta d'una imatge incorrecta que no correspongui a un abocador o que la ubicació no sigui directament en un nucli urbà, mitjançant un contenidor multimèdia que

pot alternar entre les dos vistes. També pot modificar la descripció en cas de que hi hagi faltes d'ortografia, paraules malsonants, etc. Finalment, mitjançant un *ManualWasteModal* (es veurà més endavant a l'apartat **¡Error! No se encuentra el origen de la referencia.**) l 'administrador por corretgir el tipus de residus seleccionats per a l'abocador.

Finalment, l'administrador accepta o rebutja l'abocador amb els botons inferiors, clarament diferenciats amb color vermell per rebutjar i verd per acceptar:

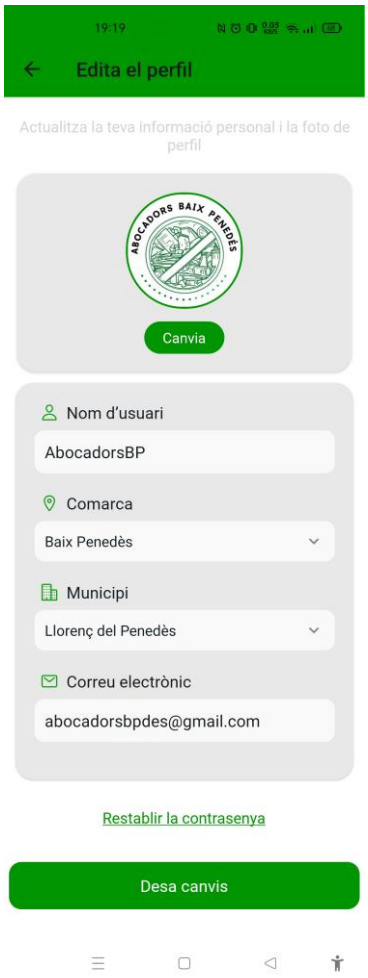


Cal remarcar que no s'ha afegit la opció d'editar la ubicació ja que l'administrador no té cap manera de saber exactament on es troba aquell abocador. Per tant, ha de confiar en l'usuari que ho ha registrat. D'aquesta manera tampoc fa falta mostrar textualment la comarca i municipi així com les coordenades de l'abocador.

D'altra banda, tampoc s'indica qui és el denunciant per evitar preferències a l'hora de gestionar abocadors.

La part final de la pantalla del perfil mostra diverses opcions relacionades amb el compte:

1. **Editar Perfil**: Aquesta pantalla constitueix l'espai dedicat a la modificació i actualització de la informació personal de l'usuari. Es prioritza la validació robusta i la sincronització bidireccional entre emmagatzematge local i servidor, garantint una experiència fluida i sense pèrdua de dades:



Il·lustració 63. Pantalla d'edició de perfil

Els inputs dels camps de text com el nom o el correu es sanititzen abans d'enviar al servidor. A més, el nom d'usuari no pot ser una cadena buida '' ni tampoc la paraula reservada 'self' ja que aquesta s'utilitza internament per fer comprovacions d'usuari. En concret per saber si s'està consultant un perfil d'usuari extern o el propi perfil de l'usuari.
En canvi els valors per comarca i municipi no s'han de sanititzar al frontend ja que es tracta de valors predefinits pel sistema que no causen errors.

La pantalla integra un botó per restablir la contrasenya, acció que requereix tenir accés al correu electrònic indicat, ja que això envia un missatge de confirmació i un correu automatitzat amb un enllaç per a restablir les credencials.

Un cop es desen els canvis, les claus d'emmagatzematge local (*AsyncStorage*) s'actualitzen i les dades s'envien al servidor per a ser processades assegurant la consistència d'aquestes.

2. **Tanca la Sessió**: Aquesta opció executa un tancament segur i complet de la sessió d'usuari mitjançant una neteja exhaustiva de totes les dades emmagatzemades localment. Quan l'usuari selecciona aquesta acció, el sistema elimina completament tot el contingut de l'emmagatzematge local del dispositiu. Aquesta neteja inclou totes les claus de dades sensibles i qualsevol altra informació relacionada amb la sessió d'usuari.

   Un cop completada la neteja, el sistema redirigeix automàticament l'usuari a la pantalla de *login* mitjançant Expo Routing de tal manera que s'evita que l'usuari pugui tornar enrere mitjançant el botó de navegació del dispositiu garantint un tancament de sessió definitiu.
   Aquest procés assegura que no quedin rastres de dades personals al dispositiu i que qualsevol intent d'accedir a pantalles protegides desencadeni automàticament una redirecció a la pantalla d'autenticació. La implementació segueix les millors pràctiques de seguretat per a aplicacions mòbils, protegint la privadesa de l'usuari i prevenint accessos no autoritzats en cas que el dispositiu sigui compartit o compromès.

3. **Eliminar Compte**: Aquesta opció implementa un procés reforçat per evitar eliminacions accidentals o no autoritzades. En seleccionar-la, s'obre un modal de confirmació que exigeix la introducció exacta del text *"Vull eliminar el meu compte AbocadorsBP"*. El sistema valida en temps real la coincidència del text, activant el botó de confirmació només quan és correcte. Aquesta mesura de seguretat garanteix que l'usuari comprengui completament la naturalesa irreversible de l'acció:

Il·lustració 64. Modal de sol·licitud d'eliminació de compte

   Durant el procés, els controls es desactiven i es mostra un indicador de càrrega, mentre que els errors es gestionen dins del mateix modal per mantenir el context.

   Un cop l'usuari ha sol·licitat l'eliminació, el servidor inicia un procés de verificació que requereix confirmació per correu electrònic.

   En cas d'èxit, l'usuari rep una confirmació visual mitjançant una alerta nativa que l'informa sobre la necessitat de revisar la seva bústia de correu per completar el procés d'eliminació. Aquest flux de doble verificació (confirmació textual + confirmació per correu) proporciona una capa addicional de seguretat i permet a l'usuari reconsiderar la decisió abans que l'eliminació sigui definitiva.

Finalment, cal esmentar que s'ha dissenyat una versió de perfil pública que s'utilitza quan un usuari consulta el perfil del denunciant d'un abocador mitjançant el mapa principal i els seus marcadors. Aquesta pantalla afegeix un *badge* indicatiu de vista pública sota el nom d'usuari i amaga les opcions de gestió de perfil i d'abocadors (per a administradors) al mateix temps que adapta les estadístiques i informació al perfil de l'usuari que s'està visitant:

Il·lustració 65. Pàgina de perfil públic d'usuari

5.3.5.4 Pantalles d'Inici de Sessió i Registre

Les pantalles d'autenticació constitueixen el punt d'accés controlat al sistema, implementant un flux que prioritza la seguretat sense comprometre la usabilitat. El disseny garanteix que només usuaris verificats puguin accedir a les funcionalitats de creació i gestió de denúncies, mantenint la integritat del sistema de reporting comunitari.

Composició de la interfície:

Les dos pantalles comparteixen els següents elements:

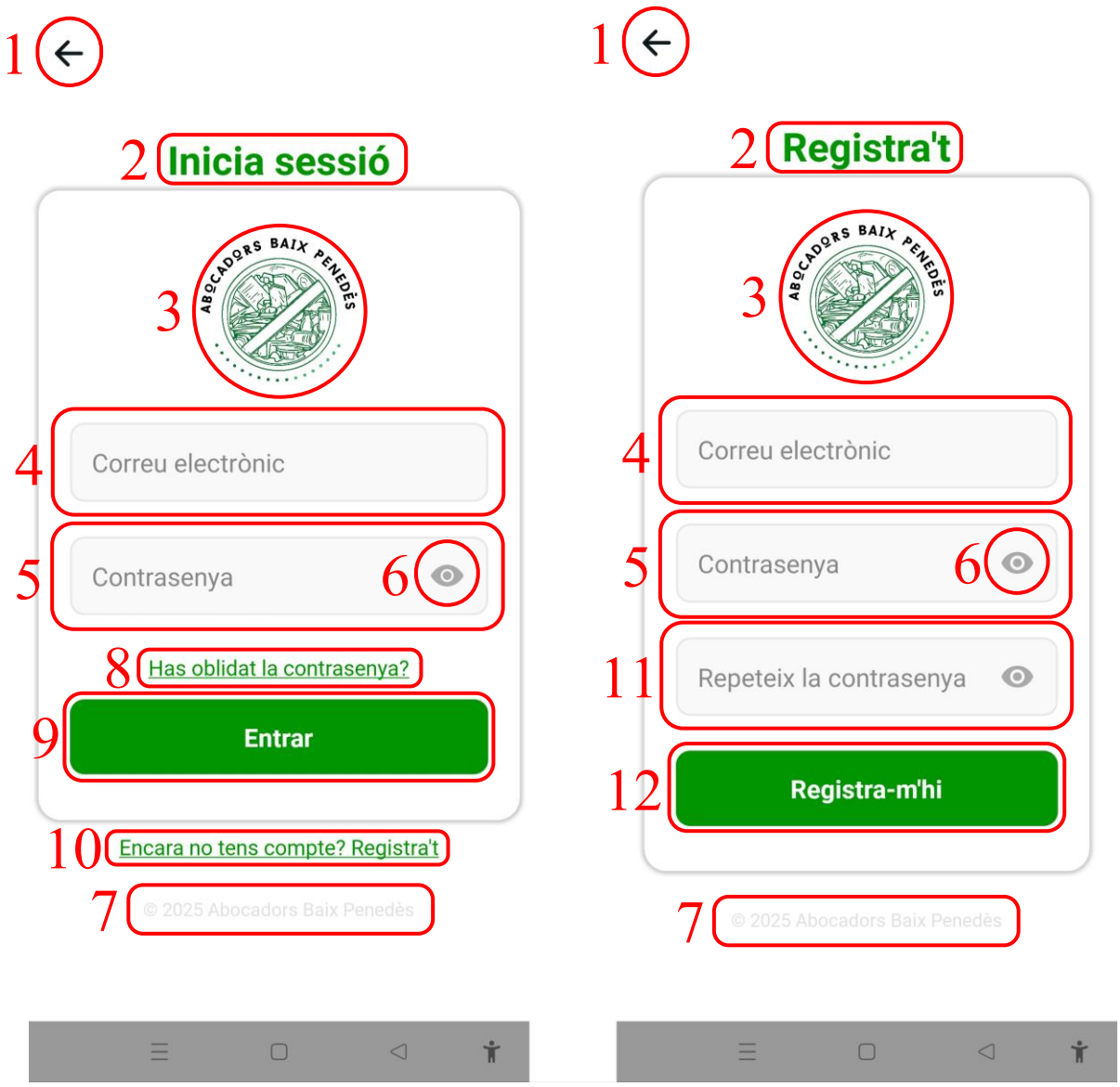
1. Botó de retrocés. En el cas de la pantalla de login retorna a l'usuari al mapa, en el cas de la pantalla de registre retorna a l'usuari a la pantalla de login.
2. Títol de la pantalla del formulari.
3. Logotip corporatiu.
4. Camp de correu electrònic.
5. Camp de contrasenya.
6. Botó per mostrar la contrasenya.
7. Peu corporatiu amb informació de copyright ©.

Elements específics de la pantalla d'inici de sessió:

8. Enllaç "Has oblidat la contrasenya?" per recuperació de credencials.
9. Botó principal "Entrar".
10. Enllaç de registre "Encara no tens compte? Registra't".

Elements específics de la pantalla del registre:

11. Camp de confirmació de contrasenya amb validació de coincidència.
12. Botó principal "Registra-m'hi".



Il·lustració 66. Pantalles d'inici de sessió i de registre amb elements numerats segons el llistat de composició.

El sistema implementa una estratègia d'autenticació basada en sessions persistents que utilitza *AsyncStorage* per mantenir l'estat d'autenticació entre inicis de l'aplicació. La presència de la clau relacionada amb l'email de l'usuari actua com a indicador principal de sessió activa, desencadenant automàticament les verificacions d'accés en pantalles protegides.

Abans de qualsevol comunicació amb el servidor, tots els inputs passen per un procés de sanitització que elimina caràcters potencialment maliciosos i normalitza les dades d'entrada. La validació del correu electrònic utilitza una expressió regular bàsica que verifica la presència d'un format vàlid amb arroba (@) i domini, mentre que les contrasenyes requereixen un mínim de 8 caràcters per establir un llindar bàsic de seguretat.

L'expressió regular que s'utilitza és molt més simple que l'estàndard RFC 5322. És una validació bàsica però efectiva per als usos de l'aplicació.

La pantalla de registre incorpora una validació addicional de coincidència entre contrasenyes, executada en temps real durant la introducció per proporcionar feedback immediat a l'usuari. Aquest enfocament preventiu redueix la frustració i accelera el procés de completació del registre.

Les peticions d'autenticació implementant gestió granular d'errors que distingeix entre diferents escenaris de fallada. Els codis de resposta 401 (credencials incorrectes), 404 (usuari inexistent), 400 (compte duplicat) i 500 (Error de connexió amb el servidor) reben tractament específic amb missatges contextuals que guien l'usuari sense revelar informació que pugui comprometre la seguretat del sistema.

Ambdues pantalles integren controls d'alternança de visibilitat per als camps de contrasenya, utilitzant iconografia universalment reconeguda (ull obert/tancat) que facilita la verificació visual de credencials sense comprometre la seguretat en entorns públics. Cal destacar que aquests botons incorporen àrees d'impacte ampliades per millorar la precisió tàctil en pantalles de menor mida o quan l'usuari porta guants, **com podria ser el cas d'un ciclista**, un perfil d'usuari que es preveu que sigui molt comú.

La funcionalitat de recuperació integrada a la pantalla de login implementa una verificació prèvia que requereix la introducció d'un correu electrònic vàlid abans de processar la sol·licitud. Aquest mecanisme preveu l'abús del sistema de recuperació i garanteix que només usuaris amb accés legítim al correu puguin iniciar el procés de restabliment.

La petició de recuperació activa un *workflow* al servidor que genera un enllaç temporal (1h) de restabliment enviat per correu electrònic, implementant una finestra de validesa limitada per motius de seguretat:


Sol·licitud de canvi de contrasenya – AbocadorsBP Safata d'entrada ×

abocadorsbpdes@gmail.com
per a mi

Hola,

Has sol·licitat canviar la contrasenya del teu compte a AbocadorsBP.

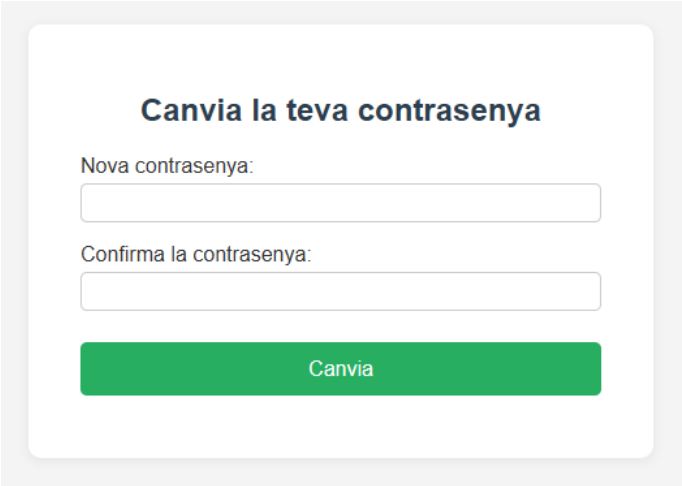
Per continuar amb el procés de canvi, si us plau, fes clic al següent enllaç: https://abocadorsbp.com/reset_password?token=VlBd05CKMrxo1y3nkPFyhDBdKjDlPa7FOxa9dEbLyvE
Aquest enllaç és vàlid durant la pròxima hora. Si no has fet aquesta sol·licitud, pots ignorar aquest missatge i la teva contrasenya actual continuarà sent vàlida.

Si tens cap dubte o problema, no dubtis a contactar amb el nostre equip de suport.

Gràcies per utilitzar AbocadorsBP!

—
L'equip d'AbocadorsBP

Il·lustració 67. Correu automatitzat de restabliment de credencials



Il·lustració 68. Formulari web de restabliment de credencials

D'altra banda, el procés de registre no activa immediatament l'accés complet al sistema, sinó que inicia un *workflow* de verificació per correu electrònic que garanteix la validesa de l'adreça proporcionada. Un cop l'usuari entra a l'enllaç de verificació del correu, se l'informa de que ja pot iniciar sessió a l'aplicació:

Il·lustració 69. Correu automatitzat per la verificació de compte registrat

Aquesta estratègia limita la creació de comptes amb correus ficticis i estableix un canal de comunicació verificat amb l'usuari.

Durant els processos d'autenticació actius, tots els elements interactius es desactiven per prevenir múltiples submissions simultànies, mentre que els botons principals exhibeixen indicadors de càrrega amb textos descriptius ("Entrant...", "Creant compte...") que mantenen l'usuari informat sobre l'estat del procés.

Finalment, els elements pressables implementen feedback visual immediat mitjançant canvis d'opacitat i escala, proporcionant confirmació tàctil que millora la percepció de responsivitat del sistema.

5.3.5.5 Components Modals Especialitzats Reutilitzables

A banda dels modals específics associats a determinades pantalles (com el d'informació d'abocador o el d'eliminació de compte, entre d'altres), l'aplicació incorpora una sèrie de **components modals especialitzats i reutilitzables**. Aquests modals s'han dissenyat com a peces independents per poder ser invocades des de diferents punts de la interfície, garantint coherència d'ús i una experiència d'usuari homogènia.

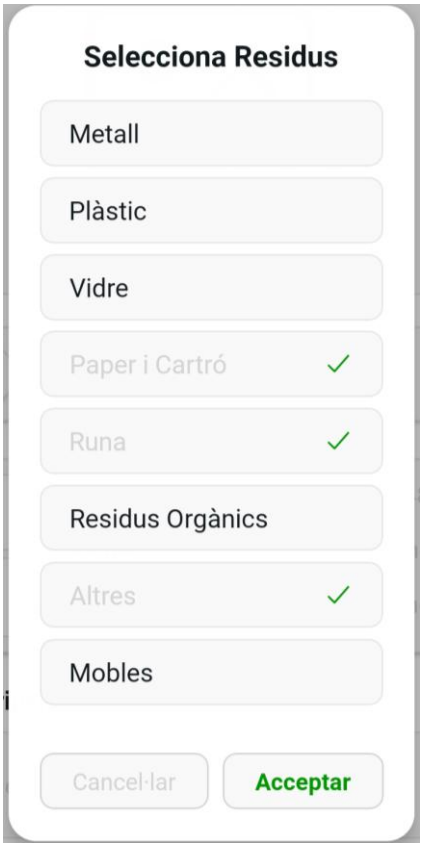
- **ImagePickerModal**: Aquest modal permet a l'usuari seleccionar una imatge des de la càmera o la galeria del dispositiu. A més de gestionar permisos, inclou opcions de retall i compressió per assegurar consistència i optimització en l'emmagatzematge. Es reutilitza en diversos contextos, com l'edició de perfil o la creació de denúncies, evitant haver d'implementar la mateixa lògica repetidament:

Il·lustració 70. ImagePickerModal

- **ManualLocationModal**: Es desplega quan l'usuari necessita introduir manualment la ubicació d'un abocador. El modal ofereix tant la selecció sobre un mapa interactiu com la introducció directa de coordenades. Aquest component s'utilitza tant en el registre de denúncies com en l'edició d'abocadors per part d'administradors, assegurant un comportament coherent en tots els fluxos on cal modificar la localització:

Il·lustració 71. ManualLocationModal

- **ManualWasteModal**: Facilita la gestió manual dels residus associats a una denúncia. Quan el model d'IA no detecta cap residu o cal afegir/ajustar categories, aquest modal permet a l'usuari seleccionar-ne d'una llista. Igual que en els casos anteriors, la seva reutilització cobreix tant la creació de denúncies com la edició i revisió d'abocadors per part dels administradors.



Il·lustració 72. ManualWasteModal

En conjunt, aquests components modals especialitzats representen un **patró de disseny reutilitzable** dins de l'aplicació: encapsulen funcionalitats freqüents però independents del context concret de cada pantalla, alhora que garanteixen una experiència uniforme i redueixen la complexitat del codi.

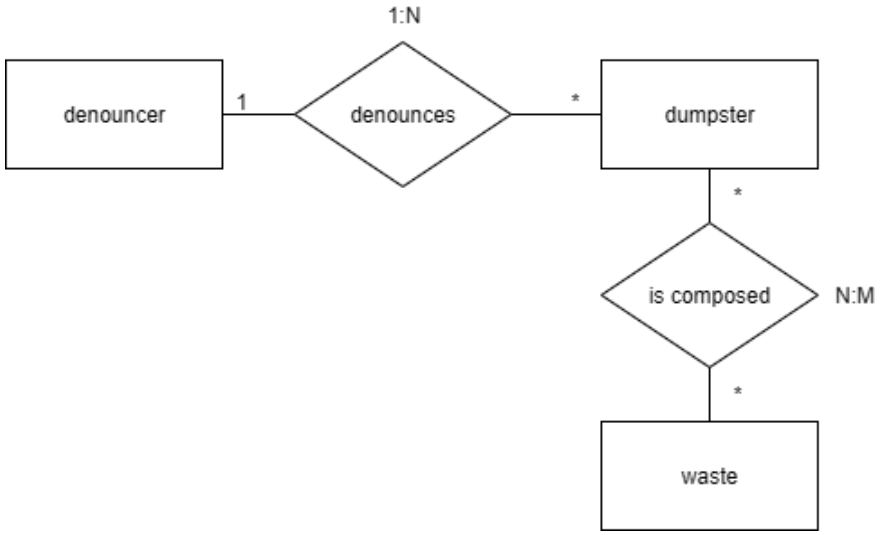
5.4 Disseny del Backend i la Persistència de Dades

5.4.1 Disseny de la Base de Dades

El disseny de la base de dades del projecte s'ha plantejat amb l'objectiu de garantir la consistència de la informació, la traçabilitat de les denúncies i la integració fluida amb la capa de backend. Per aquest motiu s'ha optat per utilitzar PostgreSQL, ja que proporciona robustesa, flexibilitat i suport avançat per a consultes i relacions complexes.

5.4.1.1 Model Entitat-Relació (MER)

El MER es centra en les entitats nuclears del domini (usuari que denuncia, abocador denunciat i llistat de residus) i en com aquestes es relacionen. Les entitats transitòries (com l'usuari pendent de verificació) reforcen el procés però no alteren l'essència del domini:



Il·lustració 73. Diagrama MER

Seguidament es fa un resum d'entitats i relacions clau:

- ***denouncer* (usuari denunciant):** Emmagatzema les dades identificatives i d'autenticació. És l'origen de les denúncies i actua també com a possible rol d'administració. La relació amb *dumpster* és **1:N**: un usuari pot crear múltiples abocadors, però cada abocador té un únic autor.pending_denouncer (usuaris pendents de verificar)
- ***pending_denouncer* (usuari pendent de verificació):** Registre temporal per al flux d'alta i verificació (token, timestamp). Separa l'estat previ a la creació de l'usuari definitiu. Conceptualment pot considerar-se entitat de procés, físicament és una taula pròpia per seguretat i higiene de dades. Aquesta taula permet evitar la creació de comptes no verificables (amb emails inexistents) assegurant que la base de dades no s'omple d'entrades inutilitzables.
- ***dumpster* (abocador).** Núcli del domini. Inclou dades d'identificació i ubicació entre d'altres. Manté una relació N:1 cap a *denouncer* per garantir la traçabilitat de l'autoria.
- ***rejected_dumpster* (abocador rebutjat).** Taula d'històric per conservar traça d'abocadors que no han estat acceptats per un administrador. Està pensada per fer-se servir en càlculs d'estadístiques, per tant, només guarda data i denunciant per no ocupar memòria excessiva a la base de dades.
- ***waste* (tipus de residu).** Catàleg controlat que sincronitza els IDs amb les classes del model YOLO. Assegura semàntica estable entre IA, API i UI.
- ***dumpster_waste* (relació N:M).** Resol la relació molts-a-molts entre *dumpster* i *waste* amb clau primaria composta per els ids d'aquestes dues entitats i amb eliminació en cascada, evitant dades orfes quan s'elimina un abocador o una categoria.

5.4.1.2 Disseny físic de la base de dades

El disseny físic de la base de dades defineix amb detall les taules, els camps i els tipus de dades que conformen l'esquema. L'objectiu ha estat reflectir fidelment el model entitat-relació i, alhora, optimitzar el rendiment i l'ús de memòria. Els tipus de dades s'han triat en funció de la naturalesa de la informació: VARCHAR i TEXT per a dades textuals, DATE per a dates, BOOLEAN per a valors lògics i FLOAT per a coordenades geogràfiques. Els identificadors únics es gestionen mitjançant SERIAL, que facilita l'autoincrement i simplifica la gestió de claus primàries.

A continuació es descriu cada taula amb els seus camps i restriccions més rellevants:

denouncer:

- **id — SERIAL, PK:** Identificador intern autoincremental.
- **pfp — TEXT:** Ruta de la foto de perfil al sistema de fitxers del servidor. El nom del fitxer està compost per el correu de l'usuari + .jpg a la carpeta *images/pfps* (p.ex: *images/pfps/abocadorsbpdes@gmail.com.jpg* → imatge de perfil de l'usuari amb correu *abocadorsbpdes@gmail.com*)
- **username — VARCHAR(100):** Àlies visible de l'usuari. Opcional i amb llargada màxima de 100 caràcters per evitar malformació d'elements a la UI.
- **region — VARCHAR(17)**: Comarca de residència de l'usuari. Opcional i amb mida màxima de 17 caràcters, ja que la comarca amb el nom més llarg és "Vallès Occidental"
- **municipality — VARCHAR(43)**: Municipi de residència de l'usuari. Opcional i amb mida màxima per 43 caràcters, ja que el municipi amb el nom més llarg és "Cruïlles, Monells i Sant Sadurní de l'Heura" [79].
- **email — VARCHAR(255), NOT NULL, UNIQUE**: Credencial principal. Obligatòria i única per evitar duplicitats.
- **paswd_hash — TEXT, NOT NULL**: *Hash* de la contrasenya.
- **reset_token — TEXT**: Token temporal per recuperació de contrasenya i eliminació de compte.
- **reset_token_expiry — TIMESTAMPTZ**: Caducitat del token de *reset*. Utilitzat per saber si ha expirat el termini de canvi de contrasenya o eliminació de compte.
- **is_admin — BOOLEAN, NOT NULL, DEFAULT FALSE**: Indicador de rol d'administrador. No compta amb cap mètode per canviar-ne el valor. Només es pot fer manualment des de la terminal de la base de dades.

pending_denouncer:

- **id — VARCHAR(64), PK**: Token de verificació (identifica el registre pendent). Els tokens són de 64 caràcters, per tant no hi ha necessitat d'alocar més memòria.
- **email — VARCHAR(255), NOT NULL, UNIQUE**: E-mail per l'alta pendent. Impedeix duplicats.
- **paswd_hash — VARCHAR(255), NOT NULL**: *Hash* de contrasenya provisional (igual que a *denouncer*).

- **created_at — TIMESTAMP, NOT NULL, DEFAULT CURRENT_TIMESTAMP**: Marca temporal de creació. Utilitzat per saber si ha expirat el termini de verificació del compte.

dumpster:

- **id — SERIAL, PK**: Identificador intern de l'abocador.
- **name — VARCHAR(11), NOT NULL, UNIQUE**: Codi públic definit com "ABP" + prefix municipal de 3 caràcters + comptador de 5 dígits. Això permet tenir nom únic per cada abocador de manera que cada un dels 947 municipis pot tenir fins a 99.999 abocadors amb nom únic (p.ex. "ABPTGN00001" → nom únic per el primer abocador registrat al municipi de Tarragona).
- **image — TEXT, NOT NULL**: Ruta de la imatge associada a l'abocador al sistema de fitxers del servidor. El nom del fitxer està compost per l'ID de l'abocador + .jpg a la carpeta *images/dumpsters* (p.ex: *images/dumpsters/1007.jpg* → imatge associada a l'abocador amb ID 1007)
- **longitude — FLOAT, NOT NULL**: Longitud geogràfica (WGS84).
- **latitude — FLOAT, NOT NULL**: Latitud geogràfica (WGS84).
- **region — VARCHAR(17), NOT NULL**: Comarca de la ubicació de l'abocador. Igual que a **denouncer**.
- **municipality — VARCHAR(43), NOT NULL**: Municipi de la ubicació de l'abocador. Igual que a *denouncer*.
- **date — DATE, NOT NULL**: Data de creació/registre de la denúncia.
- **status — VARCHAR(8), NOT NULL, DEFAULT 'pending'**: Estat de la denúncia: pending (per defecte) | accepted | retired.
- **description — TEXT, NULL**: Descripció lliure (context/notes). Opcional.
- **denouncer — INT, NOT NULL, FK → denouncer(id)**: Autor de la denúncia. Utilitzat per traçabilitat i per càlcul d'estadístiques.

Els camps de latitud i longitud es limiten a 6 posicions decimals per optimitzar l'ús de la memòria mentre que es manté una precisió teòrica de 10cm.

rejected_dumpster:

- **id — INT, PK**: Identificador original de l'abocador.
- **date — DATE, NOT NULL**: Data original de la denúncia.
- **denouncer — INT, NOT NULL, FK → denouncer(id)**: Denunciant original (per traça bàsica i estadístiques).

waste:

- **id — SERIAL, PK**: Identificador de la categoria.
- **name — VARCHAR(21), NOT NULL, UNIQUE**: Nom de la categoria.

Aquesta taula s'inicialitza amb vuit entrades on cada entrada té l'identificador de la classe de residus alineat amb els identificadors usats a l'entrenament del model d'IA: (0, Metal), (1, Plastic), (2, Glass), ...

dumpster_waste (taula d'enllaç N:M)

- **dumpster_id — INT, NOT NULL, FK → dumpster(id) ON DELETE CASCADE**: Abocador associat.
- **waste_id — INT, NOT NULL, FK → waste(id) ON DELETE CASCADE**: Categoria de residu associada.

Aquesta taula disposa d'una **clau primària composta** (dumpster_id, waste_id) que evita duplicats. L'ús de *ON DELETE CASCADE* garanteix que els enllaços s'eliminen automàticament si desapareix l'abocador o la categoria de residu.

Aquest disseny físic garanteix la coherència i integritat de les dades, alhora que proporciona una estructura clara perquè el *backend*, mitjançant l'ORM, pugui gestionar de forma eficient el cicle de vida de les denúncies.

5.4.2 Disseny de la API Backend

La API backend s'ha desenvolupat amb FastAPI, seguint els principis REST i implementant una arquitectura modular que separa clarament les responsabilitats entre validació de dades, persistència i rutes d'interacció. L'objectiu és oferir un sistema **robust, segur i escalable** que actuï com a pont entre l'aplicació mòbil, la base de dades i el model d'intel·ligència artificial.

5.4.2.1 Estructura general i separació de capes

L'arquitectura es divideix en tres capes principals:

1. **Models de dades (Pydantic):** Responsables de validar i serialitzar les entrades i sortides (longituds, formats, sanitat d'HTML, comprovació d'imatges base64).
2. **Models de persistència (SQLAlchemy):** Representen l'esquema relacional de PostgreSQL i asseguren la coherència entre taules físiques i objectes Python.
3. **Rutes (*endpoints*):** Encapsulen la lògica de negoci i connecten les capes anteriors amb la interfície mòbil i el model d'IA.

Models de Dades (Pydantic):

Per assegurar la coherència i la seguretat de les dades que circulen per la API, s'han definit diversos models base de validació (*BaseModel*) amb Pydantic. Aquests models controlen tant les entrades com les sortides, evitant la necessitat de codi repetitiu i reduint el risc d'injeccions o dades malformades.

Les validacions se centren en quatre aspectes principals:

- **Restriccions de longitud i formats**: Camps com *username*, *region* i *municipality* tenen límits màxims adaptats a la realitat administrativa (per exemple, "Vallès Occidental" o "Cruïlles, Monells i Sant Sadurní de l'Heura"). D'altra banda, *email* és obligatori i únic amb longitud màxima de 255 caràcters, mentre que *description* no pot excedir els 500 caràcters.
- **Seguretat de les dades**: Tant els textos (com noms d'usuari o descripcions) com els correus electrònics són sotmesos a sanitització HTML per prevenir injeccions, i les contrasenyes han de complir una longitud mínima de vuit caràcters.
- **Gestió d'imatges**: Les imatges de perfil i d'abocadors s'han de subministrar en format JPEG (codificat en base64), amb una mida màxima de 3.5MB (binari) ≈ 5MB un cop codificades. Aquesta restricció optimitza l'ús de memòria i assegura un format homogeni al sistema.
- **Coherència semàntica i geogràfica**: Les coordenades han de correspondre a valors vàlids dins del rang geogràfic WGS84 (*longitud*: -180..180; *latitude*: -90..90), i els residus associats a un abocador es limiten a un nombre entre 1 i 8, alineats amb el catàleg predefinit de la taula *waste*.

Amb aquests models s'han cobert els casos més habituals del domini:

- **Denouncer** (usuari): per a registres, edició de perfil i autenticació.
- **DumpsterCreate**: per a la creació d'abocadors amb coordenades i residus.
- **DenouncerLogin**: versió reduïda amb només credencials per al procés de login.
- **RejectedDumpsterCreate**: per conservar les metadades bàsiques d'un abocador rebutjat.

Aquesta estratègia permet mantenir la validació agrupada, coherent amb l'esquema físic de la base de dades i fàcilment extensible en el futur.

Models de Persistència (SQLAlchemy):

Per mapejar l'esquema físic de la base de dades a objectes Python, s'ha utilitzat SQLAlchemy com a ORM [80]. Cada classe de persistència reflecteix una taula de PostgreSQL i en defineix els camps, restriccions i relacions, de manera que la lògica de negoci pot treballar amb objectes en lloc de consultes SQL explícites.

Així, per exemple, l'entitat **DenouncerDB** representa els usuaris definitius, mentre que **PendingDenouncerDB** gestiona els usuaris pendents de verificació; **DumpsterDB** recull la informació dels abocadors registrats i es relaciona amb la taula **DumpsterWasteDB** per resoldre l'associació N:M amb els residus; finalment, **RejectedDumpsterDB** conserva la traça dels abocadors rebutjats amb finalitats estadístiques.

Aquest esquema garanteix la correspondència directa entre el model entitat-relació i el *backend*. Les consultes es realitzen de manera asíncrona mitjançant la llibreria *databases.Database*, que tradueix les operacions ORM a SQL real i assegura que les crides no siguin bloquejants. D'aquesta manera s'aconsegueix un *backend* més eficient i capaç de suportar múltiples sol·licituds concurrents sense penalitzar el rendiment.

Rutes (*endpoints*):

La capa de rutes constitueix la interfície entre l'aplicació mòbil i el *backend*. Cada *endpoint* encapsula la lògica de negoci necessària per validar les dades, interactuar amb la base de dades i retornar una resposta consistent. Tots segueixen l'estil RESTful, utilitzen JSON com a format principal d'intercanvi i retornen codis d'estat HTTP per reflectir l'èxit o el fracàs de l'operació. En casos puntuals (com la verificació de correu o la recuperació de contrasenya) es retornen respostes HTML per facilitar la interacció amb l'usuari final a través del navegador.

Per facilitar la comprensió i l'escalabilitat, les rutes s'han agrupat en blocs funcionals:

- Autenticació i gestió de compte: Registre amb verificació de correu, inici de sessió, recuperació i baixa d'usuari amb confirmació via *token*.
- Usuaris: Consulta i edició del perfil, incloent-hi la gestió de la imatge de perfil.
- Abocadors: Creació, consulta i actualització de denúncies, amb suport de paginació i filtres geogràfics.
- Administració: Acceptació, rebuig o retirada d'abocadors, així com notificacions associades.
- Estadístiques: Consultes agregades per àrea i perfils d'usuari, amb indicadors d'activitat i evolució temporal.
- Intel·ligència Artificial: classificació automàtica de residus a partir d'imatges mitjançant el model YOLO integrat al *backend*.

Aquest enfocament permet que cada bloc cobreixi un àmbit funcional concret i que els *endpoints* comparteixin convencions de disseny:

- Nomenclatura: Ús de noms en plural i verbs només en casos d'accions especials (*/accept_dumpster*, */reject_dumpster*).
- Validació i persistència: Tota entrada passa pels models Pydantic i, un cop validada, es persisteix amb SQLAlchemy.
- Tractament d'errors: S'utilitzen codis HTTP coherents (400, 401, 404, 500) amb missatges descriptius que permeten una gestió clara a la capa client.
- Paginació: Els llistats d'abocadors incorporen paràmetres *page* i *page_size*, retornant també el nombre total d'entrades per millorar l'experiència de l'usuari.

Per il·lustrar aquest disseny, a continuació es descriuen dos casos representatius:

- **POST /dumpster/register**: Permet la creació d'un abocador nou. Rep una imatge en base64, coordenades, descripció i residus associats; valida les dades, desa la imatge al sistema de fitxers i crea els registres corresponents a la base de dades. La resposta retorna l'ID de l'abocador.
- **GET /verify_email?token=…**: Completa el procés de registre d'usuari. Si el *token* és vàlid, el *backend* transfereix l'entrada de *pending_denouncer* a la taula definitiva *denouncer* i mostra una pàgina HTML de confirmació.

Altres *endpoints* segueixen patrons similars i no es descriuen per motius de seguretat.

Les decisions arquitecturals preses en el desenvolupament del backend responen a tres eixos fonamentals: escalabilitat, mantenibilitat i seguretat.

En primer lloc, s'ha adoptat un model d'execució **asíncron**, que permet gestionar de manera eficient múltiples connexions concurrents i garanteix un bon rendiment sota càrrega. En segon lloc, s'ha prioritzat la **separació de responsabilitats** amb una arquitectura modular basada en capes, fet que facilita tant l'evolució del codi com la seva reutilització i manteniment. Finalment, la **seguretat** s'ha establert com a principi transversal, incorporant mecanismes de protecció de credencials, validació estricta de dades d'entrada i gestió temporal de tokens.

Aquesta combinació proporciona una base sòlida i flexible que assegura la robustesa del sistema actual i, alhora, obre la porta a futures extensions com l'ús d'emmagatzematge d'objectes per a les imatges, l'escalat mitjançant contenidors o la integració de noves funcionalitats sense comprometre la qualitat del servei.

5.5 Disseny del Servidor

5.5.1 Infraestructura del Servidor

El *backend* del projecte s'executa en un servidor compartit llogat a la plataforma **Hetzner**, concretament el model **CX22**, amb ubicació a Falkenstein (Alemanya). Aquesta localització es va escollir perquè oferia la millor relació entre proximitat geogràfica i latència (25-35ms). Altres ubicacions disponibles (Nurmeberg) haurien incrementat el temps de resposta de les peticions (30-40ms).

El servidor utilitza **Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-64-generic x86_64)** en mode *headless* (sense entorn gràfic) i proporciona una IPv4 i IPv6 estàtiques. Tota l'administració es realitza via SSH, la qual cosa redueix el consum de recursos, simplifica el manteniment i millora la superfície de seguretat. A nivell de recursos, el CX22 ofereix 2 vCPU, 4 GB de RAM i 40 GB SSD, suficients per a la càrrega actual del sistema i escalables a instàncies superiors en cas de creixement.

5.5.2 Estructura del Projecte al Servidor

El projecte s'allotja al directori principal **AbocadorsBPAPI/**, que conté tant codi font de la API com arxius de configuració i scripts auxiliars. L'organització general és la següent:

```
AbocadorsBPAPI/
├── app/
│   ├── images/
│   │   ├── dumpsters/          # Imatges associades a abocadors
│   │   ├── pfps/               # Imatges de perfil d'usuari
│   │   └── predicting/         # Imatges temporals per inferència
│   ├── model/
│   │   ├── data.yaml           # Definició de classes del model
│   │   └── model.pt            # Pesos del model YOLO entrenat
│   ├── static/
│   │   └── privacy_policy.html # Política de privacitat servida per la API
```

```
│   ├── main.py                  # Punt d'entrada FastAPI (rutes i inicialització)
│   └── predict_yolo.py          # Càrrega i inferència amb el model YOLO
├── .env                         # Variables d'entorn (no versionat)
├── .gitattributes               # Configuració Git per fitxers grans (model.pt)
├── .gitignore                   # Exclusió de fitxers (p. ex., .env, imatges)
├── abp.geojson                  # Dades geogràfiques (àmbit del projecte)
├── abp.kml                      # Equivalent KML per eines GIS
├── comarques_municipis.json     # Diccionari de comarques i municipis
├── createDB.sql                 # Script d'inicialització d'esquema PostgreSQL
├── import_dumpsters.py          # Script de generació d'import inicial de dades
├── README.md                    # Documentació operativa bàsica
└── requirements.txt             # Dependències Python del backend
```

Rols principals per carpeta/fitxer:

- **app/main.py**: Arrenca la instància FastAPI, registra rutes i configura dependències (DB, validacions, estàtics, etc.).
- **app/predict_yolo.py**: Encapsula la càrrega del model i la inferència sobre imatges entrants (coherent amb els IDs/etiquetes de model/data.yaml).
- **app/images/**: Repositori d'imatges al servidor:
  - dumpsters/ (imatges d'abocadors, nomades per ID únic),
  - pfps/ (perfils d'usuari, nomades per correu),
  - predicting/ (temporal per fluxos d'inferència).
- **app/model/**: Contingut d'IA. Guardar model.pt al mateix host elimina dependències externes i redueix latència en inferències.
- **app/static/privacy_policy.html**: Recurs HTML servit des d'un *endpoint* per a consultes de política de privacitat o fluxos de verificació.
- **.env**: Claus i configuracions sensibles (connexió DB, secrets, límits d'imatge, etc.). No s'inclou al repositori de la URV.
- **createDB.sql**: Defineix l'esquema físic de la base de dades.
- **import_dumpsters.py**: Utilitat per carregar dades inicials i migracions puntuals.
- **abp.geojson, abp.kml, comarques_municipis.json**: Suports geogràfics i administratius per al domini (filtres, etiquetatge, validacions).
- **requirements.txt**: Llistat de dependències. Facilita desplegaments consistents.

Aquesta organització és simple i funcional: separa clarament codi, models d'IA, recursos estàtics i multimèdia, i incorpora scripts per a inicialització i import puntual. L'ús de noms deterministes per a fitxers (ID d'abocador, correu d'usuari) garanteix traçabilitat i evita col·lisions.

5.5.3 Configuració del Servidor per a Producció

El desplegament del *backend* es centra en la seguretat, la disponibilitat i la mantenibilitat. L'administració del servidor es fa via SSH amb claus públiques, evitant l'ús de contrasenyes. Les credencials i paràmetres sensibles (connexió a PostgreSQL, claus SMTP, etc.) es gestionen al fitxer *.env*, fora del control de versions. La API s'executa dins un entorn virtual Python 3.11 (venv), amb dependències controlades a *requirements.txt*.

S'utilitza **PostgreSQL 16**, configurat per acceptar només connexions locals (localhost) i autenticació per contrasenya. L'esquema es desplega amb l'script *createDB.sql*. Per mantenir la higiene de dades, s'ha configurat una tasca programada amb *cron*[62] que elimina entrades caducades de *pending_denouncer* cada hora, evitant que registres no verificats s'acumulin.

D'altra banda, el servidor FastAPI s'executa amb **Uvicorn** com a servidor ASGI[63], utilitzant (2 * nuclis_cpu) + 1 *workers* [81]. Per garantir disponibilitat contínua, Uvicorn s'ha configurat com a servei systemd, amb reinici automàtic i arrencada en boot, la qual cosa assegura que el *backend* s'executi 24/7.

Després de detectar intents d'explotació de vulnerabilitats abans de les primeres 24 hores de vida del servidor, es va implementar xifratge TLS registrant el domini **abocadorsbp.com** a la plataforma Arsys per tal de poder generar certificats amb Let's Encrypt i certbot assegurant el funcionament d'HTTPS per tot el tràfic:

```
INFO:     147.161.90.201:49148 - "GET /user_dumpster_stats/abocadorsbpdes%40gmail.com HTTP/1.1" 200 OK
INFO:     147.161.90.201:49148 - "GET /user_monthly_report_stats/abocadorsbpdes%40gmail.com HTTP/1.1" 200 OK
INFO:     147.161.90.201:49148 - "GET /user/abocadorsbpdes%40gmail.com HTTP/1.1" 200 OK
INFO:     147.161.90.201:49148 - "GET /user_dumpster_stats/abocadorsbpdes%40gmail.com HTTP/1.1" 200 OK
INFO:     147.161.90.201:49148 - "GET /user_monthly_report_stats/abocadorsbpdes%40gmail.com HTTP/1.1" 200 OK
INFO:     147.161.90.201:49148 - "GET /user/abocadorsbpdes%40gmail.com HTTP/1.1" 200 OK
INFO:     147.161.90.201:49170 - "POST /edit_user/ HTTP/1.1" 200 OK
INFO:     147.161.90.201:49170 - "GET /user_dumpster_stats/abocadorsbpdes%40gmail.com HTTP/1.1" 200 OK
INFO:     147.161.90.201:49170 - "GET /user_monthly_report_stats/abocadorsbpdes%40gmail.com HTTP/1.1" 200 OK
INFO:     147.161.90.201:49170 - "GET /user/abocadorsbpdes%40gmail.com HTTP/1.1" 200 OK
INFO:     91.92.70.10:51872 - "POST /boaform/admin/formLogin HTTP/1.1" 404 Not Found
```

Il·lustració 74. Primer intent de cerca de vulnerabilitats rebut al servidor

La seguretat de xarxa es reforça amb el Firewall de Hetzner, que es configura per permetre només connexions SSH (port 22) restringit als rangs IP de confiança (147.161.0.0/16 i 31.4.0.0/16) i connexions HTTP (port 80) i HTTPS (port 443) oberts a qualsevol IPv4 per al funcionament de la app. A més, la base de dades només escolta TCP[64] provinents de *localhost*, afegint una altre capa de seguretat.

Aquesta configuració assegura que només els serveis essencials siguin accessibles i que les connexions administratives quedin restringides a xarxes de confiança.

---

[62] cron: sistema de planificació de tasques periòdiques en entorns Unix/Linux.

[63] ASGI: Asynchronous Server Gateway Interface

[64] TCP: Transmission Control Protocol

6 Implementació

Aquest apartat explica com s'han implementat les parts més importants del projecte. El codi específic que es pugui mostrar és una simplificació del codi real.

6.1 Procés d'Entrenament del Model Classificador de Residus

En aquest apartat es detalla el procés d'entrenament, validació i integració del model d'Intel·ligència Artificial encarregat de detectar i classificar residus.

Com que es treballava amb molts fitxers multimèdia, es van començar a utilitzar barres de progrés de la llibreria **tqdm**. Aquests indicadors de progrés han ajudat a detectar errors o possibles colls d'ampolla. Per a utilitzar-los per exemple en el cas de la separació del *dataset* s'empra codi amb la següent estructura:

```
for image in tqdm(train_images, desc="Filling train_dataset"):
    ...
for image in tqdm(val_images, desc="Filling val_dataset"):
    ...
for image in tqdm(test_images, desc="Filling test_dataset"):
    ...
```

6.1.1 Preparació del Conjunt de Dades

Com bé s'ha comentat a l'apartat 5.2.3, l'entrenament del model va començar fent servir el conjunt de dades TACO, les anotacions del qual estaven en format COCO. Les anotacions en aquest format del *dataset* TACO es troben a l'arxiu `annotations.json` i tenen una estructura semblant a la següent (obviant camps innecessaris):

```
{
    "images": [
      {
         "id": 0,
          "width": 1537,
          "height": 2049,
          "flickr_640_url":
      "https://farm66.staticflickr.com/65535/3397819618_632623b4fc_z.jpg"
      },
      ...
    ],
    "annotations": [
      {
          "id": 1,
          "image_id": 0,
          "category_id": 6,
          "bbox": [
                  517,
                  127,
                  447,
```

```
                1322
            ],
        },
        ...
    ],
}
```

Sabent això es podien convertir aquestes anotacions a format YOLO. La funció `prepare_data` s'encarrega de descarregar les imatges i generar un fitxer `.txt` per a cada imatge on cada fitxer ha de contenir una fila per objecte a la imatge i on cada fila del fitxer segueix el següent format [48]:

```
class x_center y_center width height
```

Tot i que ja existeixen funcions específiques per a fer la conversió [50] , es van provar d'utilitzar però sense bons resultats, per tant es va decidir implementar la funció `convert_coco2yolo (`ubicada a l'arxiu `utils.py`). Amb un parell de línies de codi podem extreure un llistat d'anotacions que la funció convertirà a format YOLO:

```
with open("data/annotations.json", "r") as file:
    coco_data_official = json.load(file)
annotations_official = coco_data_official["annotations"]
convert_coco2yolo(annotations = annotations_official)
```

La funció doncs, comença a iterar per aquest llistat extraient per cada anotació les dades necessàries de la *bounding box* per normalitzar-les i guardar-ho tot al fitxer de text amb el format corresponent:

```
for ann in annotations:
    class = ann["category_id"]
    x, y, w, h = ann["bbox"]
    x_center = (x + w / 2) / img_width
    y_center = (y + h / 2) / img_height
    w_norm = w / img_width
    h_norm = h / img_height
    append(text_file, f"{class} {x_center} {y_center} {w_norm} {h_norm}\n")
```

Inicialment, les imatges es descarregaven del fitxer `data/all_image_urls.csv` proporcionat per TACO pensant que l'ordre d'aquestes URL[65]s d'imatges corresponia amb el camp ID de cada imatge del fitxer. Finalment es va veure que això només era cert per les primeres 100 imatges i es va canviar d'aproximació. En comptes de descarregar les imatges de `data/all_image_urls.csv`, s'obtenia l'enllaç "`flickr_640_url`" de cada imatge des de `data/annotations.json`, permetent mantenir traçabilitat de les anotacions associades a cada ID d'imatge.

Amb el *dataset* adaptat i les imatges disponible ja es podia començar l'entrenament.

[65] URL: Uniform Resource Locator

6.1.2 Reclassificació de Residus

Al cap de dues sessions d'entrenament es va identificar que la reclassificació dels residus a detectar (comentat a l'apartat 5.2.4) podria millorar els resultats del model. La solució va ser relativament fàcil i es va dividir en el següents passos, abordant el problema poc a poc:

1. Es va definir el nombre de classes (`nc`) i els seus noms a l'arxiu `data.yaml`, identificant cada classe amb un número:
```
nc: 8
names:
  0: Metal
  1: Plastic
  2: Glass
  3: Paper & Cardboard
  4: Rubble
  5: Organic Waste
  6: Other / Miscellaneous
  7: Furniture
```

2. Es va crear un diccionari amb parelles clau-valor on la clau correspon al nom de la classe original i el valor correspon a l'ID de la classe mapejada. Per exemple:
```
CLASS_MAPPING = {
    "Aluminium foil": 0,
    "Drink can": 0,
    ...
    "Garbage bag": 1,
    ...
    "Glass bottle": 2,
    ...
    "Pizza box": 3,
    ...
    "Food waste": 5,
    ...
    "Rope & strings": 6,
}
```
Es pot observar que no hi ha parelles que resolguin al valor 4 (*Rubble*) o 7 (*Furniture*) ja que TACO no disposa d'instàncies d'aquestes classes.

3. Al convertir anotacions es mapeja la classe real de cada *bounding box* mitjançant el diccionari `CLASS_MAPPING`. D'altra banda s'ha definit una "llista negra" de classes que només aporten soroll al model. Aquesta llista s'utilitza per eliminar instàncies:
```
BLACKLIST_CLASSES = [
    "Pop tab",
    "Cigarette"
]

for ann in annotations:
    class = ann["category_id"]
    if className[class] in BLACKLIST_CLASSES:
        continue
    new_class_id = CLASS_MAPPING.get(className[class])
    ...
    append(text_file,  f"{new_class_id}  {x_center}  {y_center}
{w_norm} {h_norm}\n")
```

4. Per evitar haver de re-convertir les anotacions de COCO a format YOLO i descarregar altre cop les imatges (tot això ho feia la funció `prepare_data`), es va

implementar la funció `reclassify_labels` amb la idea d'iterar per tots els fitxers `.txt` que contenen aquestes anotacions i mapejar les classes fent servir `CLASS_MAPPING`:

```
for label_file in tqdm(os.listdir(labels_dir),desc="Reclassifying
labels"):
    new_lines =  []
    with open(file_path, 'r') as f:
        lines = f.readlines()

    for line in lines:
        class_id, x_center, y_center, width, height = map(float,
line.strip().split())
        new_class_id = CLASS_MAPPING.get(className[class_id])
        new_lines.append(f"{new_class_id} {x_center} {y_center}
{w_norm} {h_norm}\n")

    label_file.writelines(new_lines)
```

Amb la reclassificació de residus acabada i la inserció de noves dades provinents d'una varietat de *datasets* de Roboflow (esmentat a l'apartat 5.2.3), es va decidir implementar una funció auxiliar `show_images_and_labels` que ajudés a visualitzar les *bounding boxes* sobre les imatges juntament amb el nom de la classe. La funció obté les coordenades de les caixes delimitadores a partir de l'arxiu `.txt` corresponent a la imatge a visualitzar i aprofitant els mòduls **pyplot** i **patches** de la llibreria **matplotlib** genera rectangles representant les caixes:

```
axes = plt.subplots(1, 1)

axes.imshow(image)

rect = patches.Rectangle((x, y), width, height)

axes.add_patch(rect)
```

El resultat és el següent:

Il·lustració 75. Resultat de la visualització personalitzada de *bounding boxes*

L'ús d'aquesta funció va proporcionar informació vital sobre la qualitat de les noves anotacions provinents dels *datasets* extrets de Roboflow.

6.1.3 Eines de Validació i Anotació

Anteriorment hem vist que TACO proporcionava 1.500 imatges oficials i unes altre 3.500 de no oficials. Afegit a això, es va estar augmentant el nostre conjunt de dades principal a base de conjunts de dades més petits i específics, la majoria provinents de Roboflow. Aquestes eines es van desenvolupar amb **tkinter** i són necessàries per a assegurar la qualitat d'aquestes dades i introduir-ne de noves ja que es va poder veure que les dades actuals contenen cert grau d'error.

Ambdues eines tenen un funcionament bàsic semblant:

Quan l'eina s'executa, el primer que fa és definir certes constants i variables globals prenent com a referència els arguments rebuts (si es que n'ha rebut) i valida que existeixin certs directoris:

Exemple Labeling Tool:

```
UNLABELED_DIR = args.src
OUTPUT_BASE_DIR = args.dst
OUTPUT_LABELS_DIR = os.path.join(OUTPUT_BASE_DIR, "labels")
OUTPUT_IMAGES_DIR = os.path.join(OUTPUT_BASE_DIR, "images")
drawing_bbox = None  # Current bounding box being drawn
bboxes = []  # List of bounding boxes for current image
undone_bboxes = []  # Stack for undo functionality
start_x = start_y = 0  # Mouse start positions
os.makedirs(OUTPUT_LABELS_DIR, exist_ok=True)
os.makedirs(OUTPUT_IMAGES_DIR, exist_ok=True)
```

Exemple Label Validation Tool:

```
LABELED_BASE_DIR = args.src
LABELED_LABELS_DIR = os.path.join(LABELED_BASE_DIR, "labels")
LABELED_IMAGES_DIR = os.path.join(LABELED_BASE_DIR, "images")
VALID_OUTPUT_BASE_DIR = args.valid
VALID_OUTPUT_LABELS_DIR = os.path.join(VALID_OUTPUT_BASE_DIR,
"labels")
VALID_OUTPUT_IMAGES_DIR = os.path.join(VALID_OUTPUT_BASE_DIR,
"images")
NONVALID_OUTPUT_BASE_DIR = args.non_valid
os.makedirs(LABELED_LABELS_DIR, exist_ok=True)
os.makedirs(LABELED_IMAGES_DIR, exist_ok=True)
os.makedirs(VALID_OUTPUT_LABELS_DIR, exist_ok=True)
os.makedirs(VALID_OUTPUT_IMAGES_DIR, exist_ok=True)
os.makedirs(NONVALID_OUTPUT_BASE_DIR, exist_ok=True)
```

Seguidament crea la finestra on es mostrarà la UI i els components d'aquesta, com els botons, que es creen mitjançant la funció `create_button` un *wrapper* que encapsula la lògica de crear un botó amb extra UI (*shortcut label* + estil):

```
def create_button(children_of, main_text, shortcut_text, command=""):
    button_container = tk.Frame(children_of, bg="#222222")
    shortcut_label = tk.Label(button_container, text=shortcut_text)
    main_button = tk.Button(button_container, text=main_text,
command=command)
    return button_container
```

Finalment l'eina enllaça les tecles del teclat amb funcions específiques a mode d'alternativa d'ús dels botons. Per passar valors a les funcions que es criden al prémer una tecla es fan servir funcions lambda:

Exemple Labeling Tool:

```
canvas.bind("<ButtonPress-1>", start_bbox)
canvas.bind("<B1-Motion>", update_bbox)
canvas.bind("<ButtonRelease-1>", finish_bbox)
canvas.bind("<Leave>", on_canvas_leave)
canvas.bind("<Motion>", update_magnifier)
root.bind('<e>', toggle_eraser_mode)
canvas.bind("<Button-3>", handle_right_click)
root.bind('<d>', toggle_dotted_lines)
root.bind('<m>', toggle_magnifier)
root.bind('<Control-z>', undo)
root.bind('<Control-Shift-Z>', redo)
root.bind('<p>', lambda event: select_image(event, previous=True))
root.bind('<n>', select_image)
root.bind('<s>', lambda event: select_image(event, save=True))
```

Exemple Label Validation Tool:

```
root.bind('<a>', lambda event: select_image(event, action='accept'))
root.bind('<r>', lambda event: select_image(event, action='reject'))
root.bind('<p>', lambda event: select_image(event, previous=True))
root.bind('<n>', select_image)
```

Per executar la Labeling Tool n'hi ha prou amb executar la comanda `python labeling_tool.py`. Addicionalment es poden passar com a arguments el directori origen (`--scr: preprocessement/unlabeled`) contenint les imatges que es volen anotar i el directori destí (`--dst: preprocessement/labeled`) on es guardaran les imatges amb les seves anotacions.

Per executar la Label Validation Tool n'hi ha prou amb executar la comanda `python label_validation_tool.py`. Addicionalment es poden passar com a arguments el directori origen (`--scr: preprocessement/labeled`), que ha de contenir dos directoris, un per les imatges (`preprocessement/labeled/images`) i un per les anotacions (`preprocessement/labeled/labels`); el directori destí per imatges validades (`--valid: data/dataset`) i el directori destí per imatges no validades (`--non_valid: preprocessement/unlabeled`).

6.1.4 Entrenament no Informat

Amb les eines funcionals i les dades finalment validades, es va procedir al primer entrenament amb el *dataset* actualitzat i completament útil (anteriorment s'havien fet entrenaments però amb resultats pèssims degut als errors que aquí ja s'havien resolt).

Primerament es va carregar el model "x" amb pesos preentrenats mitjançant la API d'Ultralytics [82]:

```
from ultralytics import YOLO
model = YOLO("yolo11x.pt")
```

Seguidament, per dividir el *dataset* en 3 *datasets* més petits (train, validation i test) es va utilitzar la funció `split_dataset` implementada a l'script `utils.py`, que desordena de manera aleatòria les imatges del conjunt de dades i les insereix a proporció 70-15-15 (per defecte, tot i que es pot indicar mitjançant arguments d'entrada) als conjunts de dades de `train`, `val` i `test` respectivament:

```
images = [f for f in os.listdir(f"{dataset_dir}/images")]
random.shuffle(images)
train_split = int(len(images) * 0.7)
val_split = int(len(images) * 0.15)
train_images = images[:train_split]
val_images = images[train_split:train_split + val_split]
test_images = images[train_split + val_split:]
...
```

Un cop tenim els diferents conjunts de dades executem un entrenament bàsic seguint la guia d'ús proporcionada per Ultralytics [83]:

```
results = model.train(
    data="data.yaml",
    epochs=100,
    batch=16,
    imgsz=416,
    device="cuda",
    model="yolo11x.pt",
)
```

Els primers entrenaments donaven errors relacionats amb CUDA[66], però després de buscar informació específica i arreglar la compatibilitat de les versions de CUDA i PyTorch es va poder solucionar el problema [84][85].

6.1.5 Hyperparameter Tuning i Entrenament Informat

Un cop acabat el primer entrenament seriós, es va intentar treure el màxim partit del model amb les dades disponibles. Després de consultar en profunditat la documentació d'Ultralytics en quant a configuracions i hiperparàmetres es va decidir quins d'aquests es podrien intentar optimitzar amb *hyperparameter tuning* [86]. Per utilitzar la integració de Ray Tune amb la API d'Ultralytics Només s'han de seguir dos passos:

1. Definir un *search space*. És a dir, un llistat dels hiperparàmetres que es volen optimitzar juntament amb el rang de valors que pot prendre cada un d'aquests a l'hora de fer la cerca. En el nostre cas només optimitzem els tres més importants:
```
search_space = {
    "lr0": tune.loguniform(1e-5, 1e-1),
    "momentum": tune.uniform(0.6, 0.98),
    "weight_decay": tune.uniform(0.0, 0.001),
}
```
2. Executar la cerca amb el *search space* definit:
```
result_grid = model.tune(
    data=os.path.abspath("data.yaml"),
    space=search_space,
    use_ray=True,
    iterations=20,
    grace_period=20,
    patience=15,
    gpu_per_trial = 1,
    optimizer = "SGD",
    device=0,
    epochs=60,
    imgsz=416,
    batch=16,
    workers=6,
    project="runs/detect",
    name="tune",
)
```

Intentar optimitzar diversos hiperparàmetres alhora provoca una explosió combinatòria en el cost computacional. En el cas d'Ultralytics, la funció `tune` fa servir Ray Tune amb una estratègia de **cerca aleatòria**, que assigna valors aleatoris dins dels rangs definits per a cada hiperparàmetre, i utilitza l'ASHA *scheduler* per aplicar *early stopping*. En el nostre experiment només s'han executat 20 iteracions, és a dir, 20 entrenaments. Com més hiperparàmetres incorporem al procés d'optimització, més gran és el nombre de combinacions possibles, i per tant cal incrementar el nombre d'iteracions per aconseguir resultats òptims. Aquí és on cal trobar un equilibri entre el nombre d'hiperparàmetres, les iteracions i el cost computacional i temporal del procés.

[66] CUDA: Compute Unified Device Architecture

Abans de poder iniciar el procés de *hyperparameter tuning* es van haver de resoldre alguns inconvenients tècnics.

En primer lloc, la cerca d'hiperparàmetres generava un error relacionat amb la longitud excessiva dels noms de directori (més de 260 caràcters) que provenien de Ray Tune. Per solucionar-ho, es va crear un *issue* al repositori oficial d'Ultralytics a GitHub després d'haver fet una recerca prèvia a internet [87]. L'endemà mateix, gràcies al suport d'Ultralytics, el problema es va poder resoldre i el procés va continuar amb normalitat.

En segon lloc, atesa la durada de les sessions d'entrenament, es va explorar la possibilitat de **combinar els recursos de la GPU i la CPU** per accelerar els experiments. Com que Ray Tune permet distribuir la càrrega de treball entre diversos dispositius (GPU o CPU), es va plantejar utilitzar Ray Tune directament sense passar per l'API d'Ultralytics [88][89][90][91]. Tanmateix, no es va trobar una solució factible i finalment es va optar per aprofitar exclusivament la GPU.

Finalment es va poder fer una primera fase d'entrenament del model amb els hiperparàmetres que van proporcionar millors resultats:

```
model = YOLO("yolo11x.pt")
results = model.train(
    data="data.yaml",
    epochs=100,
    batch=32,
    imgsz=416,
    lr0=0.0007302333570667613,
    momentum=0.9298546547609154,
    weight_decay=0.0005556272622725124,
    optimizer = "SGD",
    device="cuda",
    project=os.path.join(ROOT_DIR, "runs/detect"),
)
```

Posteriorment es va intentar fer una segona fase d'entrenament amb un *learning rate* inicial 10 cops inferior al de la primera fase, però sense millores significatives.

En resum, el procés de *hyperparameter tuning* va permetre identificar una configuració d'hiperparàmetres que millorava el rendiment del model mantenint un cost computacional raonable. Amb el model ja entrenat i llest per al seu ús, el projecte podia avançar cap al següent pas: el disseny i desenvolupament de la interfície gràfica que havia de permetre la interacció entre l'usuari i el sistema d'IA, entre altre coses.

6.2 Desenvolupament de la Interfície Gràfica

El desenvolupament de la Interfície Gràfica d'Usuari (GUI) es va iniciar amb una fase d'aprenentatge del framework React Native i de la plataforma Expo. Per assolir els coneixements bàsics es va seguir un tutorial introductori de FreeCodeCamp, [92] que va servir per comprendre l'estructura del framework i els seus components fonamentals. Posteriorment, durant aproximadament un mes, es van dur a terme diverses proves i experiments amb l'objectiu de familiaritzar-se amb les eines i establir una base sòlida. Superada aquesta etapa inicial, es va donar pas a la implementació real de l'aplicació.

6.2.1 Mapa interactiu amb Leaflet i OpenStreetMap i Comunicacions Internes

Per evitar utilitzar el component nadiu de Google Maps per React Native, com que no existeix un component nadiu de Leaflet per a React Native, es va optar per incrustar un mapa Leaflet dins de l'aplicació mitjançant un component `WebView` de React Native. En carregar la pantalla de mapa, es genera una pàgina HTML incrustada que inicialitza un mapa Leaflet i carrega les capes de mapes base desitjades.

En el fitxer `map.jsx` es defineix una constant `leafletHtml` que conté el codi HTML, CSS i JavaScript necessari per al mapa. En aquest codi s'inclouen els estils i la llibreria de Leaflet (via CDN[67]) i també la llibreria de *MarkerCluster* per agrupar els marcadors (punts de denúncia) en forma de clúster. Tot seguit, s'inicialitza el mapa definint diferents capes de mapes base. Per exemple, es configura una capa base estàndard apuntant a OpenStreetMap (`osm`), així com capes addicionals per a mapes estilitzats clar i fosc de CartoDB, i una capa satèl·lit d'Esri.

```
    const tileLayers = {
      osm:
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png'),
      light:
L.tileLayer('https://{s}.basemaps.cartocdn.com/light_all/{z}/{x}/{y}{r}.p
ng'),
      ...
      satellite:
L.tileLayer('https://server.arcgisonline.com/ArcGIS/rest/services/World_I
magery/MapServer/tile/{z}/{y}/{x}')
    };
```

La comunicació entre la part nativa (React Native) i la part WebView (Leaflet) es va implementar mitjançant missatges. React Native injecta la pàgina HTML i rep missatges dels events del mapa (com clicks o canvis de vista), i alhora pot enviar ordres al mapa (com afegir marcadors, canviar la capa base, etc.). Aquesta comunicació s'implementa amb `window.ReactNativeWebView.postMessage` al codi de la pàgina HTML i amb el prop `onMessage` de la WebView a la part de React Native:

[67] CDN: Content Delivery Network

```
<WebView
    onMessage={handleWebViewMessage}
/>
```

Per exemple, quan l'usuari prem un botó per canviar el tipus de mapa (estàndard, clar, fosc, satèl·lit), l'aplicació envia un missatge de tipus `changeMapType` a la WebView:

```
const handleMapTypeChange = useCallback((item) => {
    setMapType(item.value)
    const message = {
        type: "changeMapType",
        mapType: item.value
    }
    mapRef.current?.postMessage(JSON.stringify(message))
}, [])
```

El codi dins la WebView escolta aquest tipus de missatge i activa la capa corresponent canviant la visibilitat de les capes de Leaflet:

```
if (data.type === 'changeMapType') {
    if (tileLayers[data.mapType]) {
        map.removeLayer(currentBaseLayer);
        currentBaseLayer = tileLayers[data.mapType];
        currentBaseLayer.addTo(map);
    }
}
```

De la mateixa manera, quan es clica un marcador d'abocador al mapa, el codi HTML executa `window.ReactNativeWebView.postMessage` enviant un missatge `dumpClick` amb l'ID de l'abocador seleccionat:

```
const marker = L.marker([item.latitude, item.longitude]);
marker.on('click', function() {
    window.ReactNativeWebView.postMessage(JSON.stringify({
        type: 'dumpClick',
        dumpId: item.id
    }));
});
```

Això provoca que la part nativa obri un modal amb la informació detallada d'aquell abocador:

```
const handleWebViewMessage = useCallback((event) => {
    if (data.type === "dumpClick") {
        const dump = dumpsterDataById[data.dumpId]
        if (dump) {
            setSelectedDump(dump)
            setModalVisible(true)
```

```
            }
        }
    })
```

D'aquesta manera es defineixen totes les comunicacions entre React Native i WebView.

6.2.2 GeoJSONs, Contexts i Geometries

El fitxer GeoJSON amb geometries per les comarques [76] inicialment tenia un volum considerable però es va decidir eliminar informació inútil o redundant (com la precisió de les dades, reduint-les de 14 posicions decimals a 6; o la capital de cada comarca) mitjançant expressions regulars i eines de reducció online [93] complementades amb eines de visualització d'arxius GeoJSON per assegurar que les dades no s'havien malmès [94]. Per les geometries dels municipis [77] es va aplicar el mateix procés ja que es va haver de convertir a JSON i indentar per explorar el fitxer, entendre'n l'estructura i acomodar-lo a l'ús. A més, en aquest cas es va trobar informació duplicada que es va poder esborrar. Les geometries de comarques van passar de pesar 17.2MB a pesar 4.3MB mentre que les de municipis van passar dels 35MB als 17.4MB, rebaixant la mida inicial conjunta d'aquests dos arxius d'uns 52.2MB a 21.7MB, aproximadament un 58% de reducció, molt més acceptable per a una aplicació mòbil. Amb aquests canvis l'aplicació ja era capaç de carregar els fitxers asíncronament i servir-los com a context global `AsyncGeoDataContext`:

```
export const AsyncGeoDataProvider = ({ children }) => {
    const [comarquesData, municipisData] = await Promise.all([
        import('@/data/comarques.json'),
        import('@/data/municipis.json')
    ]);
    return (
        <AsyncGeoDataContext.Provider value={value}>
            {children}
        </AsyncGeoDataContext.Provider>
    );
};
```

Aquestes dades també es fan servir per generar llistats dels components *dropdown* amb un altre context global `ProcessedGeoDataContext`, on a més dels llistats, es demana a la API el nombre de denúncies per comarca/municipi.

L'últim context que s'utilitza és el `ThemeContext`, per establir els colors de l'aplicació, definits a `constants/Colors`:

```
const theme = colorScheme === 'dark' ? Colors.dark : Colors.light;
return (
    <ThemeContext.Provider value={{
        colorScheme, setColorScheme, theme
    }}>
```

```
        {children}
    </ThemeContext.Provider>
);
```

Aquests contexts s'apliquen a tota l'aplicació a través de l'arxiu `app/_layout.tsx`:

```
return (
    <ThemeProvider>
      <AsyncGeoDataProvider>
        <ProcessedGeoDataProvider>
          ...
        </ProcessedGeoDataProvider>
      </AsyncGeoDataProvider>
    </ThemeProvider>
```

Per enfosquir tot el mapa excepte la comarca o municipi seleccionat, es combinen dues geometries en un sol polígon amb forat.

1. Primer es carrega un GeoJSON englobant definit a data/darkBackgroundGeoJSON.json que cobreix tota la superfície. Es defineix a `data/darkBackgroundGeoJSON.json`:

```
{
    "type": "Feature",
    "geometry": {
        "type": "Polygon",
        "coordinates": [[
            [90, 360],
            [-180, -90],
            [-90, -180],
            [-360, -180],
            [180, -270],
            [90, 360]
        ]]
    }
}
```

2. En segon lloc es carrega la geometria corresponent a la selecció del filtre (comarca o municipi), que actuarà com a "forat" dins de la geometria englobant.

Al codi JavaScript de la constant `leafletHtml` es combinen aquestes dues geometries per construir un únic polígon amb un forat. D'aquesta manera, la zona seleccionada queda transparent mentre que la resta del mapa apareix semi-transparent i enfosquit:

```
if (darkPolygon && regionPolygon) {
    const hole = regionPolygon.map(poly =>
        poly.map(([lat, lng]) => [lat, lng])
    );
    const shell = darkPolygon.map(([lat, lng]) => [lat, lng]);
    const polygonWithHole = [shell, ...hole];
    L.polygon(polygonWithHole, { color: 'black', fillOpacity: 0.4,
stroke: false }).addTo(map);
```

```
}
```

Cada vegada que l'usuari canvia de comarca o municipi, el codi esborra l'anterior capa de filtre i genera un nou polygonWithHole amb la geometria corresponent.

6.2.3 Comunicació amb la API

Com s'ha vist a l'apartat 5.4.2.1, la API compta amb un seguit d'*endpoints* que l'aplicació fa servir per obtenir dades. Totes les crides que el *frontend* fa a aquestes rutes segueixen el mateix patró. Totes les crides fan servir la paraula reservada `await` per tal de fer les crides asíncronament, sense bloquejar el flux principal de l'aplicació. Seguit de la crida s'espera la resposta, també de manera asíncrona i es comproven els codis de sortida. Les crides que no facin servir el mètode GET han de tenir el mètode a usar especificat. En cas de que l'*endpoint* esperi rebre paràmetres a la ruta, aquests s'afegeixen a la crida:

```
const res = await fetch(`https://abocadorsbp.com:443/dumpster/${current.id}`, {
    method: 'DELETE',
})
const data = await res.json()
if (res.status === 200) {
    ...
```

D'altra banda, quan la ruta esperi rebre un cos JSON, aquest s'haurà d'indicar:

```
const response = await fetch(`https://abocadorsbp.com:443/register`,
{
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        email: cleanEmail,
        password: cleanPassword,
    }),
});
const data = await response.json()
if (response.status === 200) {
    ...
```

6.2.4 Visualització d'abocadors al mapa

Per tal de mantenir les imatges i dades dels abocadors del mapa a memòria, s'implementa una solució senzilla però eficaç. Un diccionari per les imatges i un per les

dades, on les claus son els IDs dels abocadors i els valors són les imatges en base64 o les dades de l'abocador:

```
const [dumpsterBase64Images, setDumpsterBase64Images] = useState({})
const [dumpsterDataById, setDumpsterDataById] = useState({})
```

Cada vegada que l'usuari mou el mapa o canvia el nivell de zoom, el codi de la pàgina Leaflet executa un `callback` que envia al component React Native un missatge de tipus `bboxChange` amb les coordenades de la nova *bounding box* visible. Aquest missatge inclou `min_lat`, `max_lat`, `min_lon` i `max_lon` corresponents als límits geogràfics de la vista actual del mapa. L'aplicació rep aquest missatge a través de la funció `handleWebViewMessage` i s'encarrega de demanar a la API els abocadors que estiguin ubicats dins aquests límits a través de la ruta `dumpster_ids_in_bbox`:

```
const idsData = fetch(`https://abocadorsbp.com:443/dumpster_ids_in_bbox?min_lat=${data.bbox.min_lat}&max_lat=${data.bbox.max_lat}&min_lon=${data.bbox.min_lon}&max_lon=${data.bbox.max_lon}&area=${encodeURIComponent(area)}`
```

Aquesta petició retorna els IDs dels abocadors ubicats dins els límits del mapa, els quals l'aplicació compara amb el diccionari de dades d'abocadors per identificar els que encara no s'han memoritzat, demanar-ne les dades a la API i mantenir la nova informació:

```
const missingIds = idsData.ids.filter(id => !dumpsterDataById[id])
const dataUrl = `https://abocadorsbp.com:443/dumpsters?` + missingIds.map(id => `ids=${id}`).join('&')
const res = await fetch(dataUrl)
const data = await res.json()
if (data && data.dumpsters) {
    setDumpsterDataById(prev => {
        const updated = { ...prev }
        data.dumpsters.forEach(dumpster => {
            updated[dumpster.id] = dumpster
        })
        return updated
    })
}
```

En paral·lel a la consulta d'IDs i dades textuals, per gestionar la càrrega d'imatges associades als abocadors visibles al mapa, s'inicia la detecció de marcadors visibles. En el codi de la WebView, després d'afegir tots els marcadors Leaflet, s'executa la funció `detectVisibleMarkers` que recorre tots els marcadors del *cluster* i determina quins són visibles individualment (és a dir, no agrupats dins d'un clúster a la vista actual):

```
if (layer instanceof L.Marker && markerClusterGroup.getVisibleParent(layer) === layer) {
    visibleMarkers.push(layer.options.dumpId);
}
```

També comprova els clústers que estan visibles per decidir si caldria "espigolar-los" (spiderfy) en cas de zoom màxim amb la funció `willClusterSpiderfy`:

```
return (
    bottomCluster._zoom === maxZoom &&
    bottomCluster._childCount === cluster._childCount &&
    markerClusterGroup.options.spiderfyOnMaxZoom
);
```

Un cop recopilada la llista d'IDs visibles (incloent-hi aquests casos especials de clúster), la funció envia un missatge a React Native de tipus requestDumpsterImage amb la llista d'IDs per als quals cal obtenir imatges. Altre cop es filtren els IDs dels abocadors que ja tenen la imatge guardada a memòria (`dumpsterBase64Images`) i es demanen les imatges al *backend* mitjançant l'*endpoint* `/dumpster_image/{id}`.

6.2.5 Emmagatzematge local amb AsyncStorage

Quan es fa login amb èxit, s'executa la funció handleLogin, que desa el correu electrònic i el flag d'administrador a AsyncStorage:

```
await AsyncStorage.setItem('@user_email', cleanEmail);
await AsyncStorage.setItem('is_admin', isAdmin ? 'true' : 'false');
```

Al tancar sessió, es fa un AsyncStorage.clear() per eliminar totes les claus guardades i assegurar que no quedi cap dada local.

La preferència de tema es gestiona des del `ThemeContext`. En canviar de tema, s'actualitza l'estat `colorScheme` i es desa a *AsyncStorage* amb la clau `@user_theme`. Quan es torna a arrencar l'app, al muntar el context es llegeix aquesta clau i es restaura el tema guardat:

```
const savedTheme = await AsyncStorage.getItem('@user_theme');
if (savedTheme) setColorScheme(savedTheme);
```

El component de perfil carrega les dades desades en muntar-se amb un `useEffect`. Es llegeixen claus com `@user_username`, `@user_region` i `@user_municipality` i s'assignen als estats locals. Si alguna clau no existeix, s'estableix un valor per defecte i es guarda immediatament a AsyncStorage:

```
useEffect(() => {
  const loadProfileData = async () => {
    const savedName = await AsyncStorage.getItem('@user_username');
    setUsername(savedName || 'Nom d\'usuari no definit');
  };
  loadProfileData();
}, []);
```

Quan l'usuari canvia la seva foto de perfil, es desa tant al servidor com localment. Amb Expo FileSystem es genera un URI de la imatge comprimida i es guarda a AsyncStorage amb la clau `@user_profile_picture_uri`. En reiniciar l'app, el component de perfil llegeix aquesta clau i mostra directament la foto local sense tornar-la a descarregar.

En resum, totes les dades persistents s'han gestionat amb AsyncStorage mitjançant claus amb el prefix `@user_` (excepte `is_admin`). La lectura i escriptura es fan sempre dins de funcions asíncrones i encapsulades en `useEffect` quan cal carregar-les en muntar els components.

6.2.6 Navegació i Altres Funcionalitats

Navegació i estructura de pantalles:

Les pantalles principals (Mapa, Càmera, Perfil) es defineixen dins el directori `app/(tabs)/`, mentre que pantalles secundàries (login, registre, gestió d'abocadors) es troben fora d'aquest grup. Per protegir rutes, s'utilitzen segments de ruta de tipus *protected*. Quan l'usuari inicia sessió, el handler de login redirigeix directament a la pantalla de perfil mitjançant `router.replace('/(tabs)/profile')`. Això substitueix la pantalla de login i evita que l'usuari pugui tornar enrere amb el botó físic. Des de la pantalla de mapa, en prémer el botó de càmera, es fa un `router.push('/(tabs)/camera')`. A cada pantalla hi ha un botó "Enrere" a la capçalera que executa `router.back()`, de manera que es conserva l'estat anterior del mapa en tornar.

Compressió d'imatges abans de pujar-les:

Abans d'enviar qualsevol foto al servidor, es crida una utilitat `compressImage` provinent del fitxer `utils/imageCompression.js` basada en Expo ImageManipulator. La funció aplica un *preset* de compressió segons el tipus d'imatge:

```
const result = await ImageManipulator.manipulateAsync(
  uri,
  [{ resize: { width: 1024, height: 768 } }],
  { compress: 0.7, format: ImageManipulator.SaveFormat.JPEG, base64: true }
);
setImage(result.uri);
setImageBase64(result.base64);
```

Mentre dura el procés, es mostra un indicador amb l'estat `compressingImage = true`.

Tant la foto de perfil com les imatges d'abocadors tenen *presets* predefinits:

```
PROFILE_PHOTO: {
    quality: 0.8,
    maxWidth: 512,
    maxHeight: 512,
    format: 'jpeg'
},
DUMPSTER_PHOTO: {
```

```
    quality: 0.7,
    maxWidth: 1024,
    maxHeight: 768,
    format: 'jpeg'
}
```

Validació de formularis i gestió d'errors

Per validar camps es van crear funcions al fitxer `utils/validation.js`. Exemples:

```
export const validateEmail = (email) =>
  /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email);

export const validateDescription = (text) =>
  text && text.length <= 500;
```

Visualització de residus

Es defineix un *array* `CLASS_NAMES` a l'arxiu `constants/WasteTypes.js` que es referència arreu del codi per transformar IDs de residus a noms amb una simple cerca:

```
import { CLASS_NAMES } from '@/constants/WasteTypes'
<Text style={styles.wasteLabel}>{CLASS_NAMES[wasteId]}</Text>
```

6.3 Implementació del Backend i la Persistència de Dades

La implementació de la API es va fer seguint la documentació de FastAPI a mode de base [95][96], així com amb l'ús de la documentació de Pydantic i SQLAlchemy.

6.3.1 Connexió i Gestió de la Base de Dades PostgreSQL

La connexió amb la base de dades es va implementar amb SQLAlchemy, definint models ORM que representen les taules principals (*denouncer*, *dumpster*, etc.). Els paràmetres de connexió es gestionen amb un fitxer `.env` separat del codi font (es tornarà a fer referència a aquest fitxer a l'apartat 6.4.2, on s'especifiquen credencials sensibles, inclosa també la contrasenya d'aplicació per a l'enviament d'email SMTP:

```
DATABASE_URL=postgresql+asyncpg://postgres:<<contrasenya_a_la_db>>@l
ocalhost:5432/AbocadorsBPDB
MAIL_USERNAME=abocadorsbpdes@gmail.com
MAIL_PASSWORD=<<application_password_per_correu>>
MAIL_FROM=abocadorsbpdes@gmail.com
MAIL_PORT=587
MAIL_SERVER=smtp.gmail.com
```

La connexió amb la DB es defineix de la següent manera:

```
DATABASE_URL = os.getenv("DATABASE_URL")
database = databases.Database(DATABASE_URL)
```

Els models ORM es defineixen heretant de Base (SQLAlchemy) [97]. Per exemple:

```
class DenouncerDB(Base):
    __tablename__ = "denouncer"
    id = Column(Integer, primary_key=True, index=True)
    = Column(String, nullable=True)
    username = Column(String, nullable=True)
    region = Column(String, nullable=True)
    municipality = Column(String, nullable=True)
    email = Column(String, unique=True, index=True)
    paswd_hash = Column(String)
    reset_token = Column(String, nullable=True)
    reset_token_expiry = Column(DateTime(timezone=True),
nullable=True)
    is_admin = Column(Integer, default=0)
```

6.3.2 Generació d'Identificadors d'Abocadors

Els abocadors registrats necessiten un identificador únic i traçable. Per a això es va preparar l'arxiu `comarques_municipis.json`, que mapeja tots els municipis de Catalunya amb la seva comarca corresponent i per cada municipi introdueix un codi d'11 caràcters (6 caràcters i 5 dígits) únic representatiu d'aquell municipi. El fitxer té la següent estructura:

```
...

"Tarragonès": [
    {
        "nom": "Altafulla",
        "id": "ABPAFA"
    },
    {
        "nom": "El Catllar",
        "id": "ABPCAT"
    },
    {
        "nom": "Constantí",
        "id": "ABPCST"
    },

    ...
],
"Baix Penedès": [

...
```

Per obtenir 947 codis representatius i que no es repetissin es va recórrer a l'ajuda de ChatGPT, ja que era massa complex d'implementar un script que generés noms **representatius** de cada municipi. Tot i la seva ajuda, encara hi havia duplicats que es van modificar manualment.

A *l'endpoint* encarregat de registrar abocadors nous (`/dumpster/register`) s'obtenen les inicials a partir del municipi i es comprovar la disponibilitat de l'ID, afegint-lo de manera incremental (ABPTGN00001, ABPTGN00002, ABPTGN00003, ...):

```
    with open(os.path.join(os.path.dirname(__file__), "..",
"comarques_municipis.json"), encoding="utf-8") as f:
        comarques = json.load(f)
    prefix = comarques.getID(dumpster.municipality)
    select_query = select(DumpsterDB).where(DumpsterDB.municipality ==
dumpster.municipality).order_by(DumpsterDB.name.desc())
    existing = await database.fetch_one(select_query)
    next_num = existint.getNumber()
    name = f"{prefix}{next_num:05d}"
```

En cas de que no es trobi el municipi a `comarques_municipis.json`, el nom de l'abocador començarà per "ABPUNK" (*unknown*):

```
    if not prefix:
        prefix = "ABPUNK"
```

6.3.3 Mecanismes de Seguretat i Robustesa

Les contrasenyes dels usuaris es guarden encriptades amb **bcrypt**. En el moment del registre, la contrasenya rebuda es transforma amb un *hash* abans de desar-la:

```
pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")
@app.post("/register")
async def register_user(user: Denouncer):
    user_dict = user.model_dump()
    pwd_hash = pwd_context.hash(user_dict["password"])
    insert_query = PendingDenouncerDB.__table__.insert().values(
        id=verify_token,
        email=user_dict["email"],
        paswd_hash=pwd_hash
    )
    await database.execute(insert_query)
```

En iniciar sessió, es valida comparant la contrasenya rebuda amb el *hash*:

```
@app.post("/login")
async def login_user(user: DenouncerLogin):
    query = DenouncerDB.__table__.select().where(DenouncerDB.email
== user.email)
    db_user = await database.fetch_one(query)
    is_valid = pwd_context.verify(user.password, db_user.paswd_hash)
    if is_valid:
        return {"code": 200, "message": "Login successful"}
    else:
        raise HTTPException(status_code=401, detail="email or
password incorrect")
```

Per validar les dades d'entrada amb Pydantic, els validadors hereten de BaseModel [98][99]. Per exemple:

```
class Denouncer(BaseModel):
    username: str | None = Field(None, max_length=100)
    pfp: str | None = Field(None, max_length=5000000)
    region: str | None = Field(None, max_length=17)
    municipality: str | None = Field(None, max_length=43)
```

```
    email: EmailStr
    password: str = Field(..., min_length=8)

    @field_validator('region')
    @classmethod
    def validate_region(cls, v):
        return html.escape(v.strip())

    ...
```

6.3.4 Consideracions en la Definició d'Endpoints

Durant la implementació es va detectar un problema amb els *endpoints* `/dumpsters` i `/dumpsters_paginated`. Si `/dumpsters_paginated` es definia després de `/dumpsters`, FastAPI l'interpretava com una crida a `/dumpsters` amb un paràmetre extra "paginated". Això és degut a que FastAPI busca *endpoints* de manera seqüencial. Per evitar-ho, els endpoints sensibles es van ordenar amb cura al codi, definint `/dumpsters_paginated` abans de `/dumpsters`:

```
@app.get("/dumpsters_paginated")
def get_pending_dumpsters(...):
    ...
@app.get("/dumpsters")
def get_dumpsters(...):
    ...
```

Això assegura que FastAPI resolgui correctament les rutes i no confongui les crides.

6.3.5 Enviament de Correus Automàtics

Les credencials SMTP es carreguen des de `.env` a l'arrencada del servidor:

```
conf = ConnectionConfig(
    MAIL_USERNAME = os.getenv("MAIL_USERNAME"),
    MAIL_PASSWORD = os.getenv("MAIL_PASSWORD"),
    MAIL_FROM = os.getenv("MAIL_FROM"),
    MAIL_PORT = int(os.getenv("MAIL_PORT", 587)),
    MAIL_SERVER = os.getenv("MAIL_SERVER"),
    MAIL_STARTTLS = True,
    MAIL_SSL_TLS = False,
    USE_CREDENTIALS = True
)
```

Per enviar un correu fem servir FastMail indicant la configuració [100]. Per exemple:

```
    verify_link=f"https://abocadorsbp.com/verify_email?token={verify_tok
en}"
    message = MessageSchema(
        subject="Verificació de correu - AbocadorsBP",
```

```
        recipients=[user_dict["email"]],
        body=f"Hola ... fes clic al següent enllaç: 🔗{verify_link} ...",
        subtype="plain"
    )
    fm = FastMail(conf)
    await fm.send_message(message)
    return {"code": 200, "message": "Correu de verificació enviat. Revisa la teva bústia per activar el compte."}
```

6.3.6 Integració amb el Model de Detecció

Per tal d'inferir les imatges i extreure'n les prediccions del model, l'endpoint `/dumpster/predict` fa una crida a la funció `predict_yolo` definida a un arxiu a part (`predict_yolo.py`). Aquesta funció s'encarrega de desar la imatge rebuda en base64 per paràmetre, ja que YOLO treballa amb imatges locals, no amb base64:

```
def predict_yolo(image_base64: str):
    temp_dir = "app/images/predicting"
    os.makedirs(temp_dir, exist_ok=True)
    img_name = os.path.join(temp_dir, f"temp_{uuid.uuid4().hex}.jpg")
    with open(img_name, "wb") as f:
        f.write(base64.b64decode(image_base64))
```

Seguidament carrega el model `app/model/model.pt` i comença la predicció:

```
    model = YOLO(MODEL_PATH)
    results = model.predict(img_name)
```

Finalment extreu els IDs dels residus detectats i elimina duplicats per posteriorment eliminar la imatge temporal i retornar els resultats:

```
    detected_wastes = [int(idx) for idx in results[0].boxes.cls.cpu().numpy()] if results[0].boxes is not None else []
    unique_wastes = list(dict.fromkeys(detected_wastes))
    os.remove(img_name)
    return unique_wastes
```

L'*endpoint* doncs envia la llista d'identificadors a l'aplicació:

```
img_results = predict_yolo(image_base64)
return {"code": 200, "message": "Residus detectats!", "detected_wastes": img_results}
```

6.4 Desplegament del Servidor

El desplegament del servidor de *backend* es va realitzar segons la planificació descrita en l'apartat 5.5, posant especial atenció a la seguretat, disponibilitat i mantenibilitat del sistema. Seguidament es detallen els passos més importants que es van seguir per desplegar el backend al servidor Hetzner així com contratemps trobats en el procés.

6.4.1 Accés Remot al Servidor

Com s'ha explicat anteriorment, la infraestructura utilitzada és un servidor virtual de Hetzner. Per tant, el primer pas després d'escollir el model que es volia fer servir (CX22) consistiria en crear una parella de claus per accedir-hi de forma remota via SSH, per evitar l'ús de contrasenyes en l'accés administratiu. Aquestes claus es van generar executant la següent comanda a un terminal Unix:

```
ssh-keygen -t ed25519 -C "el_teu_email"
```

D'aquesta manera es genera una parella de claus (clau pública i clau privada) mitjançant l'algorisme *Ed25519*, més modern, segur i eficient que RSA [101], amb un comentari d'associació de la clau amb un correu. La clau privada (id_ed25519) es va guardar a la ruta `~/.ssh/id_ed25519`. Aquesta clau no ha de ser compartida amb ningú. D'altra banda, la clau pública (id_ed25519.pub) es va afegir al servidor Hetzner per tal d'establir la connexió segura. A partir d'aaquest moment ja es pot accedir al servidor per ssh indicant amb quin usuari i a quina màquina volem connectar-nos (la IPv4 del servidor és estàtica):

```
ssh root@128.140.92.201
```

6.4.2 Preparació de l'Entorn

Un cop dins el servidor, la configuració inicial va requerir crear l'entorn virtual (`python3 -m venv .venv`) i activar-lo (`source .venv/bin/activate`) instal·lant totes les dependències descrites a `requirements.txt` (`pip install -r requirements.txt`) garantint un entorn aïllat per a l'execució de l'API.

Durant aquesta instal·lació es va detectar un problema amb la llibreria OpenCV `opencv-python`, mòdul `cv2`, que en un entorn sense interfície gràfica va provocar errors per la falta de components visuals (llibreria nativa de renderitzat libGL.so.1). Per resoldre-ho, es va substituir el paquet per la variant `opencv-python-headless`, eliminant les dependències gràfiques i permetent utilitzar OpenCV al servidor sense inconvenients:

```
deactivate
source .venv/bin/activate
pip uninstall opencv-python
pip install opencv-python-headless
```

La gestió de la configuració sensible es va fer tal com estava previst: mitjançant un fitxer de variables d'entorn `.env`. En aquest arxiu no versionat, es van definir credencials privades com les contrasenyes de la base de dades, claus d'API de tercers i altres paràmetres de configuració. D'aquesta manera, el codi font es manté separat de la informació confidencial, i l'aplicació carrega aquests valors en iniciar-se.

Seguidament, ens ubiquem al directori del projecte (`cd AbocadorsBPAPI`) i creem un arxiu anomenat `.env`. Aquest és l'arxiu que contindrà els secrets del *backend* i que editarem a mà definint variables d'entorn com:

```
DATABASE_URL=postgresql+asyncpg://postgres:<<contrasenya_a_la_db>>@l
ocalhost:5432/AbocadorsBPDB
MAIL_USERNAME=abocadorsbpdes@gmail.com
MAIL_PASSWORD=<<application_password_per_correu>>
MAIL_FROM=abocadorsbpdes@gmail.com
MAIL_PORT=587
MAIL_SERVER=smtp.gmail.com
```

Un cop preparat l'entorn, s'havia de clonar el repositori de GitHub privat que conté el codi del *backend*. Aquest repositori però es manté privat per motius de seguretat ja que conté secrets. Per tant, per poder clonar-lo es va haver de generar un PAT[68]. Els PATs es generen des de la configuració de perfil de GitHub. un procés que no porta més de cinc minuts.

Amb el PAT actiu, es va poder clonar el repositori, on es va executar la següent comanda que requeria especificar usuari (HappyMistery) i *token* (PAT) per funcionar:

```
git clone https://github.com/HappyMistery/AbocadorsBPAPI.git
```

Això va desplegar tots els components necessaris, incloent-hi els scripts d'inicialització com `createDB.sql` per crear l'esquema de la base de dades i altres utilitats definides durant el disseny.

La base de dades PostgreSQL (versió 16) es va instal·lar i configurar seguint els criteris de seguretat establerts. Primer es va accedir a l'usuari (`sudo -iu postgres`) i executant la comanda `psql` es va accedir al client de PostgreSQL. Un cop dins, per tal de modificar la contrasenya de l'usuari es va executar la següent comanda:

```
ALTER ROLE postgres WITH PASSWORD '<<nova_contrasenya>>';
```

A més, també es va aprofitar per crear la base de dades:

```
CREATE DATABASE "AbocadorsBPDB" OWNER postgres;
```

[68] PAT: Personal Access Token

Seguidament es van fer dues modificacions per tal de que PostgreSQL pogués rebre peticions de `localhost`, ja que per defecte, PostgreSQL a Debian/Ubuntu no escolta TCP:

1. Es va descomentar la següent línia del fitxer `/etc/postgresql/16/main/postgresql.conf`:

   ```
   # listen_addresses = 'localhost'
   ```

2. Es va modificar el fitxer `/etc/postgresql/16/main/pg_hba.conf` per a que hi hagués una línia com la següent:

   ```
   # TYPE   DATABASE    USER      ADDRESS       METHOD
   Host       all       all     127.0.0.1/32     md5
   ```

Després d'aplicar aquests canvis era necessari reiniciar el servei amb la comanda `systemctl restart postgresql`.

A partir d'aquest moment, per accedir a la nostre base de dades s'ha d'indicar el nom d'aquesta juntament amb l'usuari i la contrasenya:

```
psql
postgresql://postgres:<<contrasenya_a_la_db>>@localhost:5432/AbocadorsBPD
B
```

Finalment, en provar el funcionament del servidor mitjançant *uvicorn* tot funcionava:

```
(.venv) root@AbocadorsBPServer:~/AbocadorsBPAPI# uvicorn
app.main:app --host 0.0.0.0 --port 80
INFO:     Started server process [33645]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:80 (Press CTRL+C to
quit)
```

Davant aquest èxit, es va procedir a aplicar l'script `createDB.sql` per generar l'esquema de taules i relacions definit al disseny(`\i ~/AbocadorsBPAPI/createDB.sql`).

Un cop fet tot això, l'entorn base ja estava preparat.

6.4.3 Seguretat i Disponibilitat

Com s'ha comentat a l'apartat 5.5.3, en les primeres 24 hores de servei es van detectar intents de cerca de vulnerabilitats des d'internet, fet que va plantejar l'ús d'HTTPS per a les comunicacions.

Per afegir HTTPS és necessari disposar d'un certificat SSL/TLS que es pot obtenir mitjançant Let's Encrypt [102][103], però això requereix d'un domini registrat, cosa que fins ara el servidor no tenia, sinó que treballava amb IPv4 pública estàtica. Així doncs, es va registrar un domini a la plataforma d'Arsys (abocadorsbp.com i

www.abocadorsbp.com) on posteriorment es van configurar les entrades DNS per a que resolgui les rutes registrades a la IPv4 i la IPv6 del servidor:

Entrada DNS	Tipo	Valor
abocadorsbp.com	A	128.140.92.201
abocadorsbp.com	AAAA	2a01:4f8:c013:f511::1
www.abocadorsbp.com	A	128.140.92.201
www.abocadorsbp.com	AAAA	2a01:4f8:c013:f511::1

Il·lustració 76. Resolució de rutes del domini abocadorsbp a adreces IPv4 i IPv6 del servidor

Un cop es va tenir el domini registrat, es va instal·lar *certbot* al servidor per generar el certificat associat al domini:

```
sudo apt install certbot -y
sudo certbot certonly --standalone -d abocadorsbp.com -d
www.abocadorsbp.com
```

Amb el certificat generat, només va fer falta passar-lo com a paràmetre a l'hora d'iniciar el servidor així com indicar el port 443 (port per defecte d'HTTPS):

```
uvicorn app.main:app --host 0.0.0.0 --port 443 --ssl-certfile
/etc/letsencrypt/live/abocadorsbp.com/fullchain.pem --ssl-
keyfile  /etc/letsencrypt/live/abocadorsbp.com/privkey.pem --workers 5
```

Arribats a aquest punt tot funcionava correctament, però el servidor Uvicorn només s'executava mentre hi havia una connexió ssh activa amb el servidor Hetzner, per tant, per garantir la disponibilitat contínua del *backend*, es va crear un servei dedicat amb **systemd** per a Uvicorn (`sudo vi /etc/systemd/system/abocadorsbp.service`). Aquest servei s'inicia automàticament en cada reinici del servidor i està configurat per rellançar el procés de l'API en cas de caiguda inesperada. Això assegura un funcionament ininterromput del *backend* (operatiu 24/7) sense necessitat d'intervenció manual. El codi del servei es pot trobar als Annexes.

Un cop definit el servei, es va activar i iniciar:

```
sudo systemctl daemon-reexec
sudo systemctl daemon-reload
sudo systemctl enable abocadorsbp
sudo systemctl start abocadorsbp
```

6.4.4 Inserció Inicial de Dades

Un cop el servidor estava operatiu i l'esquema de la base de dades creat, el pas següent va ser omplir aquesta base de dades amb les dades inicials del projecte. Per a això es va utilitzar un script Python anomenat `import_dumpsters.py`, desenvolupat prèviament per automatitzar la càrrega massiva d'informació. Aquest script llegeix un fitxer en format KML que conté la ubicació i dades descriptives d'una col·lecció inicial d'abocadors il·legals (aproximadament uns 1080) [104], i per a cada abocador realitza tres accions principals: en primer lloc desa una imatge representativa de l'abocador al sistema de fitxers del servidor (al directori `app/images/dumpsters` del projecte), en segon lloc passa aquesta imatge al model de IA i n'extreu els residus predits i, en tercer lloc, genera un fitxer SQL amb totes les instruccions INSERT necessàries per incorporar aquests abocadors i les seves dades a la base de dades de manera automàtica (`import_all_dumpsters.sql`).

Aquest script però es va executar localment (a l'entorn de desenvolupament) de manera que el servidor ja tingués l'script SQL present. Així doncs només faltaven les imatges, que primerament es van comprimir de la mateixa manera que ho farien si s'enviessin des de l'aplicació: utilitzant la llibreria **Pillow** amb una compressió del 70% de qualitat, mantenint un bon balanç entre qualitat visual i mida de fitxer. El codi utilitzat per la compressió es pot trobar als Annexes. Després de comprimir-les es van transferir fent servir **scp** (Secure Copy) sobre ssh, ubicant-les a les rutes corresponents dins del directori del projecte:

```
scp -r "E:\Jaume\AbocadorsBPAPI\app\images\dumpsters"
root@abocadorsbp.com:/root/AbocadorsBPAPI/app/images/
```

També es van haver de transferir els pesos del model `model.pt` per *scp* degut a un error inesperat que semblava indicar que no s'havia descarregat bé al fer `git pull`:

```
scp "E:\Jaume\AbocadorsBPAPI\app\model\model.pt"
root@abocadorsbp.com:/root/AbocadorsBPAPI/app/model/model.pt
```

6.4.5 Implementacions Adicionals

Finalment, per evitar l'acumulació de comptes d'usuari no verificats, es va programar una tasca automàtica de manteniment de dades. Un procés **cron** que s'executa cada hora i revisa la taula de registres pendents de validació (`pending_denouncer`). En concret la instrucció a afegir va ser la següent:

```
0 * * * * sudo -u postgres psql -d AbocadorsBPDB -c "DELETE FROM
pending_denouncer WHERE created_at < NOW() - INTERVAL '1 hour';"
```

Aquells usuaris que no hagin completat la verificació del seu compte en el temps establert són eliminats de la base de dades de manera segura, optimitzant l'ús de la DB.

7 Avaluació

En aquesta secció s'avalua el grau de compliment dels objectius del projecte amb especial atenció als requisits no funcionals. Primer es presenta la precisió del model d'IA (cronologia d'entrenaments YOLOv11 i mètriques) i, tot seguit, com s'han satisfet la resta de requisits no funcionals, verificant que el sistema compleix els estàndards de qualitat i és útil en un entorn real.

7.1 Precisió del model d'Intel·ligència Artificial

L'objectiu principal de era garantir que el detector ajudés més del que perjudiqués, encertant una proporció significativa de residus amb pocs falsos positius. Per validar aquest requisit es van dur a terme **27 entrenaments successius** (anomenats *train*, *train1*, ..., *train26*, ubicats a la carpeta `runs/detect/` del projecte). A continuació s'explica la cronologia:

- **Entrenaments inicials (*train* – *train3*):** El primer entrenament (***train***), amb 1.500 imatges i anotacions desalineades (1.400), presenta un rendiment nul. En corregir les anotacions (***train1***, mateix volum de dades), s'assoleix una línia base funcional (**20.7% mAP@50–95; 24.8% mAP@50; 34.3% P; 21.3% R**). Afegir *momentum* (***train2***) no va aporta millores (**18.8% mAP@50–95**), però va estabilitzar la corba (menys oscil·lació entre èpoques). Ampliant lleument la quantitat de dades i èpoques (***train3***) el model puja moderadament (**28.8% mAP@50–95; 34.0% mAP@50**).

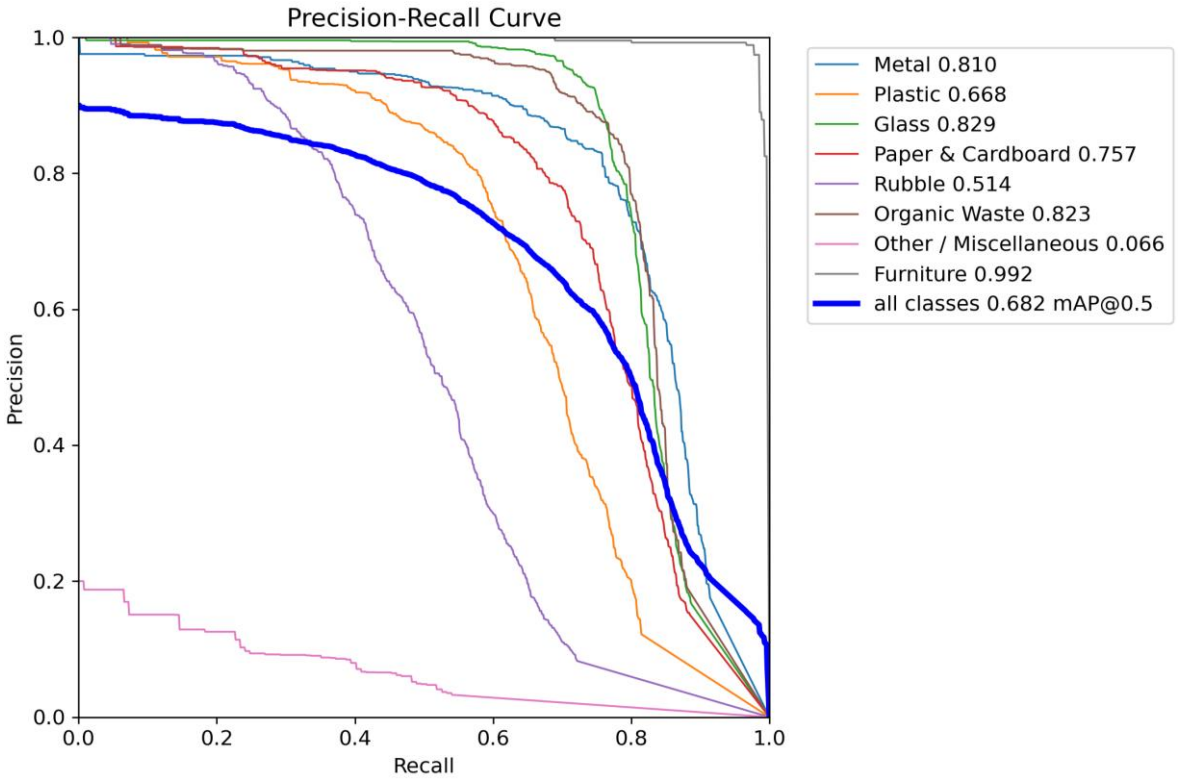
- **Primer entrenament vàlid (*train4*):** El canvi determinant arriba amb el primer entrenament sobre el conjunt ampliat (~13k imatges, train4). El model fa un salt fins a **51.9% mAP@50–95; 65.6% mAP@50; 71.4% P; 60.3% R**, demostrant l'impacte decisiu de la qualitat i l'escala del conjunt de dades i **considerant-se útil**.

- **Fase d'optimització d'hiperparàmetres (*train5–train24*):** Sobre el mateix *dataset* gran, es van executar diversos ***trials*** d'hiperparàmetres (***train5–train24***). El millor assaig **(*train22*)** assoleix **55.2% mAP@50–95; 68.4% mAP@50; 76.1% P; 62.0% R**. Aquest patró suggereix un detector conservador (poques deteccions errònies però també alguns objectes que s'escapen), sovint associat a **classes desequilibrades**.

- **Entrenament consolidat (*train25*, *train26*):** Amb els hiperparàmetres del millor trial es va fer un entrenament "net" en dues fases. La **primera fase (train25, 100 èpoques, batch 32)** reprodueix essencialment el sostre de ***train22*** (**55.1% mAP@50–95; 68.2% mAP@50; 74.5% P; 62.1% R**), confirmant la robustesa dels resultats.
  La **segona fase (*train26*)**,un *fine-tuning* amb *learning rate* inicial 10× més baix durant 50 èpoques, **no aporta millora**: el millor punt apareix molt aviat (època 2/50) i queda pràcticament idèntic a train22/train25 (**55.2% mAP@50–95; 68.4% mAP@50; 74.4% P; 62.1% R**). Això indica que el model ja havia arribat a un *plateau* de precisió amb la informació i l'equilibri de classes actuals, i un entrenament més prolongat amb els mateixos paràmetres no aportava millores substancials.

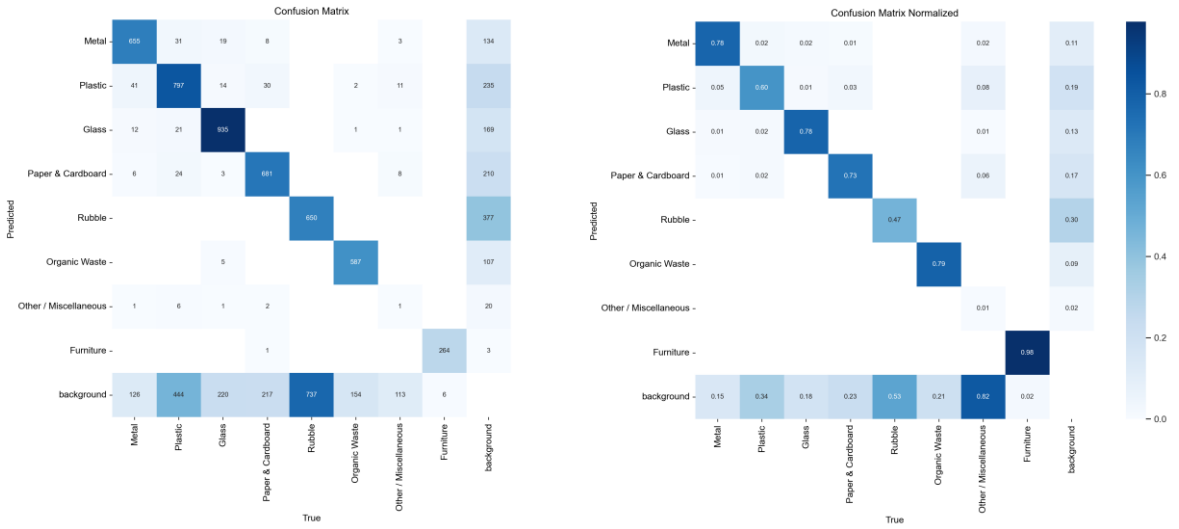
Atès als resultats, el model que s'usarà en producció correspon a la configuració millor valorada (equivalent a *train25*).

Les corbes **Precision–Recall** per classe indiquen un rendiment alt i consistent en **Mobles/Furniture** (AP@0.5 ≈ **0,99**), **Vidre** (≈ **0,83**) i **Orgànic** (≈ **0,82**), seguides de **Metall** (≈ **0,81**) i **Paper & Cartró** (≈ **0,76**). **Plàstic** queda en un rang mitjà (≈ **0,67**), mentre que **Runes** baixa fins a valors moderats (≈ **0,51**). La classe **Altres/Misc** presenta un resultat baix (≈ **0,07**), concordant amb la seva naturalesa heterogènia i poc representada:



Il·lustració 77. Gràfica *Precission-Recall Curve* de *train25*

A les **matrius de confusió** S'observa una diagonal predominant, reflex de bones prediccions en general. No obstant, hi ha confusions notables: p. ex., només ~47% dels objectes de tipus **Runa** (*Rubble*) es detecten correctament com a tal, mentre que un ~21% es confonen amb **Orgànic** i un ~30% ni tan sols es detecten (predits com a fons). En canvi, classes com **Mobles** tenen un encert ~98% i confusió mínima:



Il·lustració 78. Matrius de confusió (crua i normalitzada) de train25

El procés iteratiu ha fet evolucionar un model inicial inservible (**mAP ≈ 0 %**) fins a un model que supera el **55 % de mAP@50–95**, és a dir, que detecta i classifica correctament més de la meitat dels residus en imatges de prova amb criteris d'IoU exigents. Amb el llindar de confiança recomanat (~**0,35–0,40**), s'obté **F1 ≈ 0,67** i, per exemple, amb *recall* ≈ **0,5** la **precisió** es manté **> 0,8**; això confirma que el sistema aporta valor real a l'usuari.

Tot i que sempre és possible seguir refinant (més dades, augmentacions dirigides o arquitectures futures com YOLOv12), dins l'abast d'aquest projecte el rendiment obtingut compleix les expectatives inicials de **precisió**.

7.2 Compliment de la resta de Requisits No Funcionals

Pel que fa a la resta d'atributs de qualitat, el sistema garanteix la **privacitat** mitjançant **HTTPS/TLS** (Let's Encrypt) i evita l'exposició de dades personals; la informació sensible es conserva en una base de dades interna no accessible des d'Internet. La **seguretat** es reforça amb sanititzación i validació d'entrades al client i al servidor, **contrasenyes amb hash** (Passlib) i gestió de secrets mitjançant fitxers `.env`; l'accés administratiu es limita a **SSH amb claus** i *firewall* restrictiu, la BD només escolta en `localhost` i els **rols** es verifiquen al backend en cada acció privilegiada.

En **rendiment**, l'app aplica *lazy loading* i memòria cau, redueix els **GeoJSON un 58%** i comprimeix imatges abans d'enviar-les; al servidor, **FastAPI+Uvicorn** amb múltiples *workers* **i** latències ~25–35 ms, assegurant respostes àgils. La **fiabilitat** es garanteix executant el backend com a servei **systemd** (arrencada automàtica i *autorestart*), amb **tasques `cron`** de manteniment i **logs** per diagnòstic continu. En **compatibilitat**, l'app React Native/Expo s'ha provat en diversos Android i iOS, amb UI **responsiva** i **tema clar/fosc**, i una base de codi **portable** a Web. En el moment de publicació d'aquest document, l'aplicació es troba en fase de validació amb proves internes i externes tant a Android com a iOS. Finalment, la **mantenibilitat i escalabilitat** es fonamenten en una arquitectura **modular en capes** i tipatge (Pydantic, SQLAlchemy, etc).

En conjunt, el sistema **compleix** els requisits no funcionals: és **robust, segur, eficient i respectuós amb la privacitat**, i presenta una base sòlida per créixer sense comprometre l'experiència d'usuari.

8 Conclusions

Aquest TFG ha materialitzat **AbocadorsBP**, una plataforma completa (app mòbil, API i mòdul d'IA) per a la detecció i el registre d'abocadors il·legals, amb vocació d'ús real i col·laboració amb administracions locals del Baix Penedès. El projecte no s'ha quedat en un prototip sinó que s'ha desplegat en servidor propi amb domini, xifrat TLS i servei 24/7, cosa que ha obligat a treballar amb criteris de seguretat, privacitat i mantenibilitat propis d'un entorn de producció. A més, s'ha arribat fins al punt de publicar l'aplicació a la Google Play Store (en fase de prova tancada amb uns 30 testers de l'entorn proper) i a la App Store d'iOS (de moment amb accés públic però amb previsió de restringir-lo).

Des del punt de vista tècnic, s'ha integrat React Native (Expo) amb una API FastAPI i PostgreSQL, aprofitant validació de dades, ORM i documentació OpenAPI. La configuració sensible s'ha externalitzat via variables d'entorn, la base de dades s'ha restringit a `localhost`, i l'API corre amb Uvicorn gestionat per systemd, amb certificat Let's Encrypt sobre domini propi. Aquest conjunt ha aportat coneixements d'arquitectura, persistència i DevOps en un cas real i operatiu.

En intel·ligència artificial, el model YOLOv11 ha passat d'un inici inservible a un rendiment sostingut útil per al domini, després de 27 entrenaments i una fase d'*hyperparameter tuning*: al voltant de **55% mAP@50–95** (train25), amb patró de classes fort en Mobles/Vidre/Orgànic i més feble en *Altres* i *Runa*. A nivell de decisió de producte, això es tradueix en un detector que "ajuda més que molesta" i que orienta clarament les següents prioritats de dades.

Aquest treball ha actuat com a pont entre teoria i pràctica del grau. Anàlisi i disseny de sistemes, desenvolupament mòbil (tot i que en el meu cas específic no ha format part del meu recorregut acadèmic), aprenentatge automàtic i gestió de bases de dades, tot integrat en un cicle de vida complet (dades → entrenament → validació → integració → desplegament). S'ha après a prendre decisions tecnològiques justificades (p. ex. FastAPI+Python per integrar IA), a dissenyar esquemes i models ORM coherents, i a operar un *backend* segur i disponible. També s'han hagut de resoldre incidències de producció i a gestionar claus, PATs i certificats en un entorn real.

He constatat un sostre de rendiment amb les dades actuals i desequilibris de classe que penalitzen Runa i Altres. El camí de millora és clar: augment i reequilibri de dades, *active learning*, i avaluació de noves variants (p. ex. la següent generació de YOLO). Paral·lelament, la comunicació amb ajuntaments i consells comarcals serà fonamental un cop es disposi de dades consistents per donar-los valor. Aquest TFG, per tant, deixa una base sòlida i una agenda de treball prioritzada.

8.1 Consideracions Ètiques

Més enllà dels aspectes tècnics, aquest projecte incorpora una dimensió ètica que cal posar en relleu. La detecció i denúncia d'abocadors il·legals mitjançant una aplicació mòbil contribueix al bé comú i a la protecció del medi ambient, entesos com a patrimoni col·lectiu que cal preservar. La implicació ciutadana esdevé així una eina d'empoderament social ja

que no sols permet agilitzar la vigilància ambiental, sinó que fomenta la corresponsabilitat i la conscienciació col·lectiva en relació amb la cura del territori.

Alhora, la gestió de dades personals sensibles, com la geolocalització o les imatges capturades pels usuaris, s'ha abordat des d'una perspectiva de privacitat i seguretat, respectant la normativa vigent i aplicant el principi de minimització de dades. L'objectiu és garantir que la informació recollida només serveixi per a la finalitat prevista i que es protegeixi de manera adequada mitjançant mecanismes d'anonimització, encriptació i privacitat des del disseny.

Pel que fa a la intel·ligència artificial, el model de detecció ha estat plantejat amb criteris de transparència i supervisió humana, de manera que les seves prediccions siguin interpretables i verificables. La responsabilitat última de les decisions recau sempre en les persones, i la IA es concep com un suport.

Amb això veiem que la tecnologia no és neutral, sinó que transmet valors i ha de dissenyar-se amb un compromís clar amb la sostenibilitat, la responsabilitat social i la confiança pública. Així, AbocadorsBP no només és una eina tecnològica, sinó també una iniciativa que integra ètica i sostenibilitat en el seu nucli.

En síntesi, el TFG ha estat clau per a la meva formació. He consolidat l'enginyeria *end-to-end* d'un producte amb IA i GIS, des del codi fins a la posada en producció, amb resultats mesurables i utilitat social tangible. Aquesta experiència em prepara per afrontar projectes reals amb criteri tècnic, autonomia i responsabilitat.

9 Recursos Utilitzats

9.1 Programari

9.1.1 Llenguatges de programació

- Python (v3.13.3) - Llenguatge principal del desenvolupament i l'entrenament del model de IA i del Backend de l'API
- JavaScript/Node.js (v22.16.0) – Llenguatge principal del frontend
- TypeScript (v19.0.10) - Superset tipat de JavaScript per millor desenvolupament
- JSX/TSX - Sintaxi d'extensió per React per crear interfícies d'usuari
- JSON/GeoJSON
- SQL - Llenguatge per a la gestió i consultes de la base de dades PostgreSQL
- HTML/CSS

9.1.2 Frameworks i llibreries de Machine Learning / Deep Learning

- torch (v2.6.0+cu118) - Framework de deep learning principal
- torchvision (v0.21.0+cu118) - Llibreria de visió per computador per PyTorch
- ultralytics (v8.3.100) - Framework YOLO per detecció d'objectes
- numpy (v2.1.1) - Computació numèrica
- ray[tune] - Hyperparameter tuning
- albumentations - Augmentació d'imatges

9.1.3 Frameworks per Backend

- FastAPI 0.116.1 - Framework web modern per crear APIs REST amb Python
- Uvicorn 0.35.0 - Servidor ASGI per executar aplicacions FastAPI

9.1.4 Frameworks per Frontend Mòbil

- React (v19.0.0) - Framework principal per crear interfícies d'usuari
- React Native (v0.79.4) - Framework per desenvolupament d'aplicacions mòbils natives
- Expo (v53.0.13) - Plataforma per desenvolupament ràpid d'aplicacions React Native
- Expo Router (v5.1.1) - Sistema de navegació basat en fitxers per aplicacions Expo

9.1.5 Altres biblioteques i utilitats

Desenvolupament i entrenament del model de IA:

- opencv-python (v4.11.0.86) - Processament d'imatges
- matplotlib (v3.10.1) - Visualització de dades
- pandas (v2.2.3) - Manipulació de dades
- requests (v2.32.3) - Peticions HTTP
- tqdm (v4.67.1) - Barres de progrés
- pillow (v11.1.0) - Processament d'imatges
- natsort - Ordenació natural

- os - Operacions del sistema operatiu
- json - Processament de JSON
- time - Funcions de temps
- shutil - Operacions d'arxius d'alt nivell
- random - Generació de números aleatoris
- argparse - Parsejat d'arguments de línia de comandes
- tkinter - Interfície gràfica d'usuari
- importlib - Importació dinàmica de mòduls

Frontend:

- @expo/vector-icons (v14.1.0) - Col·lecció d'icones vectorials per aplicacions Expo
- @react-native-async-storage/async-storage (v2.2.0) - Emmagatzematge asíncron persistent per React Native
- @react-navigation/bottom-tabs (v7.3.10) - Navegació amb pestanyes inferiors
- @react-navigation/native (v7.1.6) - Llibreria de navegació per React Native
- expo-location (v18.1.6) - API per accedir a la ubicació del dispositiu
- expo-image-picker (v16.1.4) - Selecció d'imatges des de la galeria o càmera
- expo-image-manipulator (v13.1.7) - Manipulació i edició d'imatges
- expo-haptics (v14.1.4) - Retroalimentació hàptica i vibracions
- react-native-gesture-handler (v2.27.2) - Gestió avançada de gestos tàctils
- react-native-reanimated (v3.17.4) - Animacions fluides i performants
- react-native-safe-area-context (v5.4.0) - Gestió d'àrees segures en diferents dispositius
- react-native-element-dropdown (v2.12.4) - Component dropdown personalitzable
- react-native-webview (v13.13.5) - Component per mostrar contingut web dins l'aplicació

Backend:

- Pydantic (v2.11.7) - Validació de dades i serialització amb tipus de Python
- SQLAlchemy (v2.0.41) - ORM per a la gestió de bases de dades
- Databases (v0.9.0) - Biblioteca per connexions asíncrones a bases de dades
- AsyncPG (v0.30.0) - Driver asíncron per a connexions PostgreSQL
- Passlib (v1.7.4) - Biblioteca per hash i verificació de contrasenyes
- BCrypt (v3.2.2) - Algoritme de hash segur per contrasenyes
- FastAPI-Mail (v1.5.0) - Biblioteca per enviar correus electrònics des de FastAPI
- Ultralytics (v8.3.100) - Framework per models YOLO de computer vision
- Python-dotenv (v1.1.1) - Carrega variables d'entorn des de fitxers .env
- Cryptography (v45.0.5) - Biblioteca criptogràfica per seguretat
- Pillow (PIL) (v11.1.0) - Biblioteca per processament d'imatges
- Requests (v2.32.3) - Biblioteca per realitzar peticions HTTP

9.1.6 Entorns de desenvolupament

- VS Code - Editor de codi principal

Desenvolupament i entrenament del model de IA:

- Jupyter Notebook - Entorn interactiu (arxius .ipynb)
- tensorboard - Visualització de mètriques d'entrenament
- IPython - Shell interactiu de Python (utilitzat en notebooks)

Frontend:

- ESLint (v9.25.0) - Eina de linting per mantenir qualitat del codi
- eslint-config-expo (v9.2.0) - Configuració predefinida d'ESLint per projectes Expo
- Metro - Bundler integrat per aplicacions React Native/Expo

9.1.7 Control de versions

- Git
- GitHub

9.1.8 Altres eines web

Frontend:

- EAS (Expo Application Services) - Plataforma per construcció i desplegament d'aplicacions
- Expo Dev Client (v5.2.4) - Client de desenvolupament per proves en dispositius reals
- Leaflet - Biblioteca JavaScript per mapes interactius
- OpenStreetMap - Proveïdor de tiles de mapa

Backend:

- Base64 - Codificació per manejar imatges en format text (mòdul estàndard de Python)
- PostgreSQL - Sistema de gestió de bases de dades relacionals

UML:

- LucidChart – Eina de creació de diagrames UML
- Draw.io - Eina de creació de diagrames UML

9.2 Hardware

A continuació es detallen els dispositius utilitzats durant el desenvolupament del projecte, així com el seu ús específic en diferents fases com el desenvolupament i entrenament del model de detecció, el desenvolupament del frontend i backend, i les proves de l'aplicació mòbil:

OMEN 30L Desktop GT13 (PC):

- Processador: AMD Ryzen 5 5600X 6-Core Processor @ 3.70 GHz
- RAM: 16 GB
- GPU: NVIDIA GeForce RTX 3060 (12 GB)
- Sistema operatiu: Windows 11 Pro versió 24H2

HP Laptop 15:

- Processador: Intel® Core™ i5-10210U CPU @ 1.60 GHz
- RAM: 16 GB
- GPU: Intel® UHD Graphics (128 MB)
- Sistema operatiu: Windows 10 Home versió 22H2

Realme 6 Pro:

- Processador: Qualcomm® Snapdragon™ 720G Octa-core
- RAM: 8 GB
- GPU: Adreno 618
- Sistema Operatiu: Android 11

HUAWEI P30 Lite:

- Processador: HiSilcon Kirin 710
- RAM: 4 GB
- GPU: Mali-G51 MP4
- Sistema Operatiu: Android 9.0 Pie – EMUI 12.0.0

iPhone XR:

- Processador: Apple A12 Bionic
- RAM: 3 GB
- GPU: Sense nom – GPU de 4 nuclis del A12 Bionic
- Sistema Operatiu: iOS 16.0 (20A362)

Seguidament es detalla una taula per facilitar la visualització de l'ús de cada dispositiu:

	Desenvolupament i entrenament del model YOLO	**Desenvolupament frontend**	**Desenvolupament backend**	**Proves de l'aplicació**
OMEN 30L Desktop GT13	✓	✓	✓	✓
HP Laptop 15		✓		
Realme 6 Pro				✓
HUAWEI P30 Lite				✓
iPhone XR				✓

Taula 6. Taula d'ús dels diferents dispositius

Referències

[1] IPCC https://report.ipcc.ch/ar6syr/headline.html. [Canvi climàtic al 2023] 12-06-25

[2] For Tomorrow https://www.fortomorrow.eu/en/blog/ipcc2022-part1. [Resum IPCC 6th Assessment Report 2022] 12-06-25

[3] WRI https://www.wri.org/insights/ipcc-report-2022-climate-impacts-adaptation-vulnerability. [Anàlisi d'impactes i vulnerabilitats del canvi climàtic segons l'IPCC] 12-06-25

[4] UNEP https://www.unep.org/resources/emissions-gap-report-2024. [Informe d'emissions 2024] 12-06-25

[5] The Times https://www.thetimes.com/uk/environment/article/climate-change-action-worse-after-emissions-hit-record-high-n8pcprg75. [Augment de les emissions globals i inacció política]. 12-06-25

[6] Financial Times https://www.ft.com/content/ed16d880-4f63-4d64-b62e-5d2681a59494. [Paper clau del G20 en la reducció d'emissions globals] 12-06-25

[7] URV https://www.urv.cat/media/upload/arxius/normatives/propia/activitat_universitaria/investigacio/norm_prop_intelectual.pdf. [Normativa sobre propietat industrial i intel·lectual de la Universitat Rovira i Virgili] 16-08-25

[8] Ultralytics https://docs.ultralytics.com/es/models/yolo11/. [Ultralytics YOLOv11] 07-02-25

[9] SPOTTERON https://www.spotteron.net/apps. [Apps civils per al medi ambient] 03-02-25

[10] Expo https://expo.dev/. [Tauler d'inici Expo | Expo.dev] 14-06-25

[11] FastAPI https://fastapi.tiangolo.com/tutorial/. [Tutorial bàsic FastAPI] 14-06-25

[12] Leaflet https://leafletjs.com/. [Ús bàsic mínim de Leaflet] 21-06-25

[13] Wikipedia https://es.wikipedia.org/wiki/Protocolo_para_transferencia_simple_de_correo. [Wikipedia SMTP] 12-08-25

[14] Wikipedia https://es.wikipedia.org/wiki/Red_neuronal_residual. [Wikipedia ResNet] 21-01-25

[15] arXiv https://arxiv.org/abs/1512.03385. [Paper oficial ResNet] 21-01-25

[16] Wikipedia https://en.wikipedia.org/wiki/MobileNet. [Wikipedia MobileNet] 21-01-25

[17] arXiv https://arxiv.org/abs/1704.04861. [Paper oficial MobileNet] 21-01-25

[18] Wikipedia https://en.wikipedia.org/wiki/EfficientNet [Wikipedia EfficientNet] 21-01-25

[19] arXiv https://arxiv.org/abs/1905.11946 [Paper oficial EfficientNet] 21-01-25

[20] ML Journey https://python-bloggers.com/2020/10/getting-started-with-image-classification-fastai-resnet-mobilenet-and-more/. [Classificació d'imatges: Comparació entre ResNet, MobileNet i EfficientNet] 21-01-25

[21] Coursera https://www.coursera.org/articles/object-detection-vs-image-classification. [Diferència entre Object Detection i imatge Classification] 21-01-25

[22] Educative https://www.educative.io/answers/object-detection-vs-image-classification. [Detecció d'Objectes vs Classificació d'imatges] 21-01-25

[23] Medium https://medium.com/@abhishekjainindore24/object-classification-vs-object-localization-vs-object-detection-caf74860f473. [Object Classification vs Object Localization vs Object Detection] 21-01-25

[24] arXiv https://arxiv.org/abs/1911.09070. [Paper oficial EfficientDet] 21-01-25

[25] GitHub https://github.com/google/automl/blob/master/efficientdet/README.md. [Pàgina de Github d'EfficientDet] 27-01-25

[26] GeeksForGeeks https://www.geeksforgeeks.org/computer-vision/how-single-shot-detector-ssd-works/. [Com funcionen els Single-Shot-Detectors (SSD)?] 21-01-25

[27] arXiv https://arxiv.org/abs/1512.02325. [Paper oficial SSD] 21-01-25

[28] arXiv https://arxiv.org/abs/2005.12872. [Paper oficial DETR] 21-01-25

[29] arXiv https://arxiv.org/abs/1506.01497. [Paper oficial Faster R-CNN] 21-01-25

[30] KeyLabs https://keylabs.ai/blog/yolov8-vs-faster-r-cnn-a-comparative-analysis/. [Yolov8 vs Faster R-CNN] 21-01-25

[31] Medium https://medium.com/suitable-ryans-thoughts/yolov8-vs-efficientdet-a-deep-dive-into-the-future-of-real-time-object-detection-code-at-bottom-86513de2f69d. [Yolov8 vs EfficientDet] 21-01-25

[32] Medium https://medium.com/cord-tech/yolov8-for-object-detection-explained-practical-example-23920f77f66a. [Explicació de YOLOv8 per detecció d'objectes] 21-01-25

[33] Ultralytics https://docs.ultralytics.com/integrations/ray-tune/. [RayTune a YOLO] 09-04-25

[34] Ultralytics https://docs.ultralytics.com/models/yolo12/. [Ultralytics YOLOv12] 18-08-25

[35] Medium https://medium.com/@maksymilian.pilzys/cross-platform-development-showdown-flutterflow-vs-react-native-vs-flutter-for-mvps-0306c4743e97. [FlutterFlow vs React Nativa vs Flutter per MVPs] 13-04-25

[36] Expo https://docs.expo.dev/versions/latest/. [Expo APIs] 14-04-25

[37] Pydantic https://docs.pydantic.dev/latest/. [Documentació Pydantic] 23-06-25

[38] PostGIS https://postgis.net/. [Informació sobre PostGIS. Què és?] 25-06-25

[39] GeeksForGeeks https://www.geeksforgeeks.org/mysql/difference-between-mysql-and-postgresql/. [PostgreSQL vs MySQL] 24-06-25

[40] IamHectorotero https://iamhectorotero.github.io/yolo/. [Funcionament de YOLO] 22-01-25

[41] V7Labs https://www.v7labs.com/blog/yolo-object-detection. [Arquitectura YOLO] 22-01-25

[42] Datacamp https://www.datacamp.com/blog/yolo-object-detection-explained. [Explicació YOLO per detecció d'objectes] 24-01-25

[43] arXiv https://arxiv.org/abs/1506.02640. [Paper oficial YOLO] 22-01-25

[44] Ultralytics https://docs.ultralytics.com/compare/yolo11-vs-yolov8/#performance-head-to-head-yolo11-vs-yolov8. [Comparació entre models YOLO] 07-06-25

[45] LearnOpenCV https://learnopencv.com/yolo11/. [Millores de YOLOv11] 10-02-25

[46] GitHub https://github.com/garythung/trashnet. [GitHub oficial Trashnet + Conjunt de dades] 01-02-25

[47] Kaggle https://www.kaggle.com/datasets/feyzazkefe/trashnet. [Trashnet data set] 01-02-25

[48] Ultralytics https://docs.ultralytics.com/datasets/detect/. [Estructura ideal de conjunt de dades per YOLO] 01-02-25

[49] DeepWiki https://deepwiki.com/sbsepul/Repetitive-Archetypes-Patterns-Dataset/2.3-coco-to-yolo-conversion. [Procés de conversió d'anotacions COCO a format YOLO] 03-04-25

[50] GitHub https://github.com/ultralytics/JSON2YOLO.git. [Convertidor JSON2YOLO d'Ultralytics] 03-04-25

[51] Medium https://medium.com/red-buffer/converting-a-custom-dataset-from-coco-format-to-yolo-format-6d98a4fd43fc. [Convertir COCO annotations a YOLO Labels] 03-04-25

[52] GitHub https://github.com/pedropro/TACO?tab=readme-ov-file. [README TACO *dataset*] 28-03-25

[53] Encord https://encord.com/blog/taco-dataset-guide/. [Anàlisis de dades del *dataset* TACO] 28-03-25

[54] Roboflow https://universe.roboflow.com/material-identification/garbage-classification-3. [Dataset de residus quotidians] 07-04-25

[55] Roboflow https://universe.roboflow.com/surface-debris-detection-7g1df/test-debrsi. [Dataset de runa d'obra 1] 07-04-25

[56] Roboflow https://universe.roboflow.com/peter-ttwu3/concrete-detection. [Dataset de runa d'obra 2] 07-04-25

[57] Roboflow https://universe.roboflow.com/bigfootbot/furniture-cfmja. [Dataset de mobles 1] 07-04-25

[58] Roboflow https://universe.roboflow.com/minoj-selvaraj/furniture-sfocl. [Dataset de mobles 2] 07-04-25

[59] Roboflow https://universe.roboflow.com/roboflow-100/furniture-ngpea. [Dataset de mobles 3] 07-04-25

[60] GeeksForGeeks https://www.geeksforgeeks.org/machine-learning/handling-imbalanced-data-for-classification/. [Classes desequilibrades en un conjunt de dades] 07-04-25

[61] GeeksForGeeks https://www.geeksforgeeks.org/machine-learning/ml-introduction-to-transfer-learning/. [Què és el *Transfer Learning*?] 11-04-25

[62] Microsoft https://learn.microsoft.com/en-us/windows/ai/fine-tuning. [Conceptes de fine-tuning de models] 29-04-25, 17:50

[63] Ultralytics https://docs.ultralytics.com/integrations/albumentations/. [Data Augmentation d'Ultralytics amb Albumentation] 18-04-25

[64] GeeksForGeeks https://www.geeksforgeeks.org/machine-learning/hyperparameter-tuning/. [Què és *hyperparameter tuning*?] 09-04-25

[65] Ultralytics https://docs.ultralytics.com/es/guides/hyperparameter-tuning/. [YOLO hyperparameter tuning] 09-04-25

[66] Ultralytics https://docs.ultralytics.com/integrations/ray-tune/. [YOLO & Ray Tune] 09-04-25

[67] Medium https://medium.com/data-scientists-diary/stochastic-gradient-descent-sgd-and-adam-4fe496ef1bbf. [SGD i Adam] 12-04-25

[68] MassedCompute https://massedcompute.com/faq-answers/?question=What%20are%20the%20key%20differences%20between%20Adam%20and%20SGD%20optimizers%20in%20large%20language%20model%20training?. [Diferències entre SGD i Adam] 12-04-25

[69] Ultralytics https://docs.ultralytics.com/modes/train/#train-settings. [Paràmetres d'entrenament – Early Stopping (*patience*)] 10-04-25

[70] Ultralytics https://docs.ultralytics.com/guides/yolo-performance-metrics/. [Mètriques de rendiment de YOLO] 20-04-25

[71] ATTRACT GROUP https://attractgroup.com/blog/expo-vs-react-native-cli-choosing-the-right-tool-for-your-mobile-app-development/. [Expo vs React Native CLI] 21-05-25

[72] Medium https://medium.com/@manangadhiya/react-native-cli-vs-expo-in-2025-which-one-should-you-choose-326adadee590. [Escollir entre Expo i React Native CLI] 21-05-25

[73] Expo https://docs.expo.dev/build/introduction/. [Introducció a EAS Build d'Expo] 27-05-25

[74] Expo https://docs.expo.dev/router/introduction/. [Introducció a Expo Router] 21-05-25

[75] Medium https://medium.com/@madhavsharma.dev/dynamic-text-scaling-in-react-native-with-tailwind-size-matters-6342403f36e7. [Llibreria Size Matters a React Native] 01-08-25

[76] GitHub https://github.com/gloriamacia/comarques-catalunya/blob/479c7bc033edc101323a468a89dac4575378a2f3/amb-capital/data/catalunya_comarques.geojson. [fitxer .GeoJSON de Comarques de Catalunya] 21-06-25

[77] GitHub https://github.com/ArnauInes/geometries_cat_bcn_2024/blob/d08318d565dfd99b466bc89233a23ca746553cad/dts_municipis_cat_2025.json. [fitxer .GeoJSON de Municipis de Catalunya] 04-07-25

[78] Wikipedia https://es.wikipedia.org/wiki/OpenStreetMap. [Explicació OpenStreetMap] 16-06-25

[79] La Razón https://www.larazon.es/cataluna/este-municipio-nombre-mas-largo-cataluna_20241007670402a56bab1600010b13a6.html. [Municipi amb el nom més llarg de Catalunya] 26-07-25

[80] SQLAlchemy https://docs.sqlalchemy.org/en/20/orm/. [Documentació de SQLAlchemy ORM] 11-07-25

[81] Gunicorn https://docs.gunicorn.org/en/stable/design.html - how-many-workers. [Com decidir la quantitat de workers a uvicorn] 25-07-25

[82] Ultralytics https://docs.ultralytics.com/reference/data/converter/. [informació de ultralytics/data/converter.py] 03-04-25, 22:25

[83] Ultralytics https://docs.ultralytics.com/usage/python/. [Ús de YOLOv11 a Python] 07-02-25

[84] Ultralytics https://docs.ultralytics.com/guides/yolo-common-issues/. [YOLO Troubleshooting] 05-04-25

[85] PyTorch https://pytorch.org/get-started/locally/. [Compatibilitat entre versió PyTorch i CUDA] 05-04-25

[86] Ultralytics https://docs.ultralytics.com/usage/cfg/#train-settings. [Configuracions i hiperparàmetres de YOLO] 10-04-25

[87] GitHub https://github.com/ultralytics/ultralytics/issues/20101#issue-2984157779. [GitHub Issue per error de noms de directoris massa llargs al utilitzar Ray Tune amb YOLO] 12-04-25

[88] Ray https://docs.ray.io/en/latest/tune/tutorials/tune-resources.html. [Guia de paral·lelisme i recursos per Ray Tune] 20-04-25

[89] Ray https://docs.ray.io/en/latest/train/user-guides/hyperparameter-optimization.html - train-tune. [Hyperparameter Tuning amb Ray Tune] 20-04-25

[90] Ray https://docs.ray.io/en/latest/tune/tutorials/tune-run.html. [Executar experiments bàsics de Tune] 20-04-25

[91] Ray https://docs.ray.io/en/latest/tune/api/doc/ray.tune.with_resources.html. [Funció wrapper per especificar recursos - ray.tune.with_resources] 20-04-25

[92] Youtube https://www.youtube.com/watch?v=sm5Y7Vtuihg. [Tutorial bàsic de React Native per freeCodeCamp] 05-05-25

[93] ReduceGeoJSON https://reducegeojson.radicaldata.org/. [reductor d'arxius GeoJSON] 05-07-25

[94] Mapshaper https://mapshaper.org/. [Visualitzador d'arxius .GeoJSON] 05-07-25

[95] FastAPI https://fastapi.tiangolo.com/#installation. [Tutorial bàsic de FastAPI] 09-07-25

[96] FastAPI https://fastapi.tiangolo.com/tutorial/. [Guia d'usuari de FastAPI] 09-07-25

[97] SQLAlchemy https://docs.sqlalchemy.org/en/20/orm/quickstart.html. [Introducció ràpida a SQLAlchemy] 10-07-25

[98] Pydantic https://docs.pydantic.dev/latest/api/base_model/. [Documentació de Pydantic - BaseModel] 11-07-25

[99] Pydantic https://docs.pydantic.dev/latest/concepts/validators/. [Validació de camps amb field_validator de Pydantic] 22-07-25

[100] GeeksForGeeks https://www.geeksforgeeks.org/python/sending-email-using-fastapi-framework-in-python/. [Enviar email automàtic amb fastAPI] 14-07-25

[101] GeeksForGeeks https://www.geeksforgeeks.org/devops/rsa-vs-ed25519-which-key-pair-is-right-for-your-security-needs/. [Ed25519 vs RSA] 05-07-25

[102] Medium https://medium.com/@mariovanrooij/adding-https-to-fastapi-ad5e0f9e084e. [Guia per afegir HTTPS a FastAPI] 23-07-25

[103] Softwarecosmos https://softwarecosmos.com/how-to-convert-http-to-https/. [Guia per passar d'HTTP a HTTPS] 25-07-25

[104] Google Maps https://www.google.com/maps/d/u/5/viewer?hl=ca&ll=41.26173492295338%2C1.512220313752594 7&mid=1Tk5PJ2d-LnADcJYlf_8WQavZjsQ9Luk&z=11. [mapa públic d'AbocadorsBP] 09-07-25

Annexes

IA i Mètriques

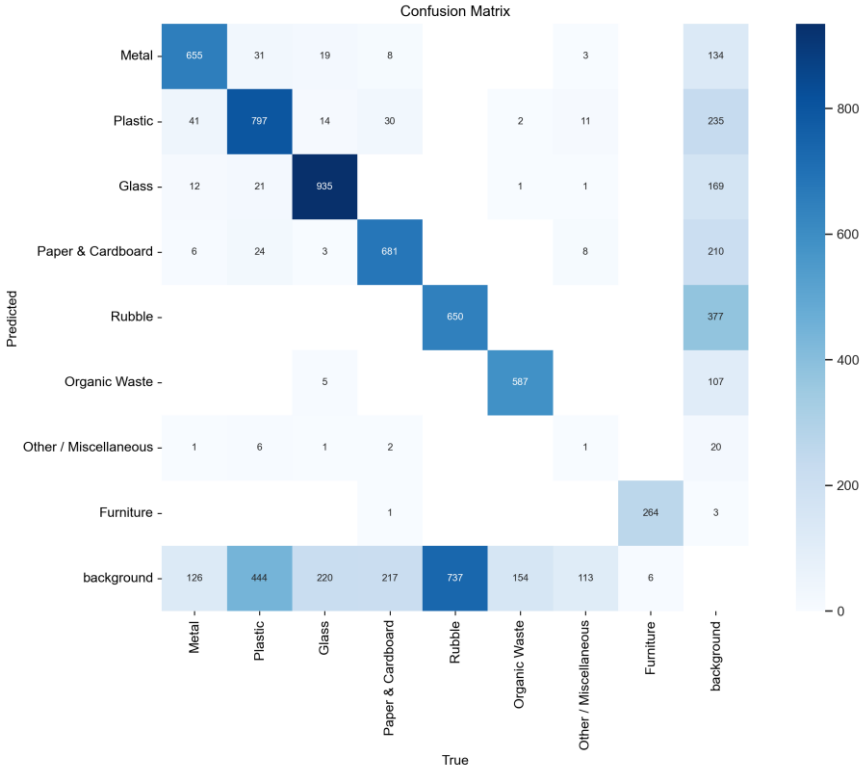
Taula resum d'entrenaments

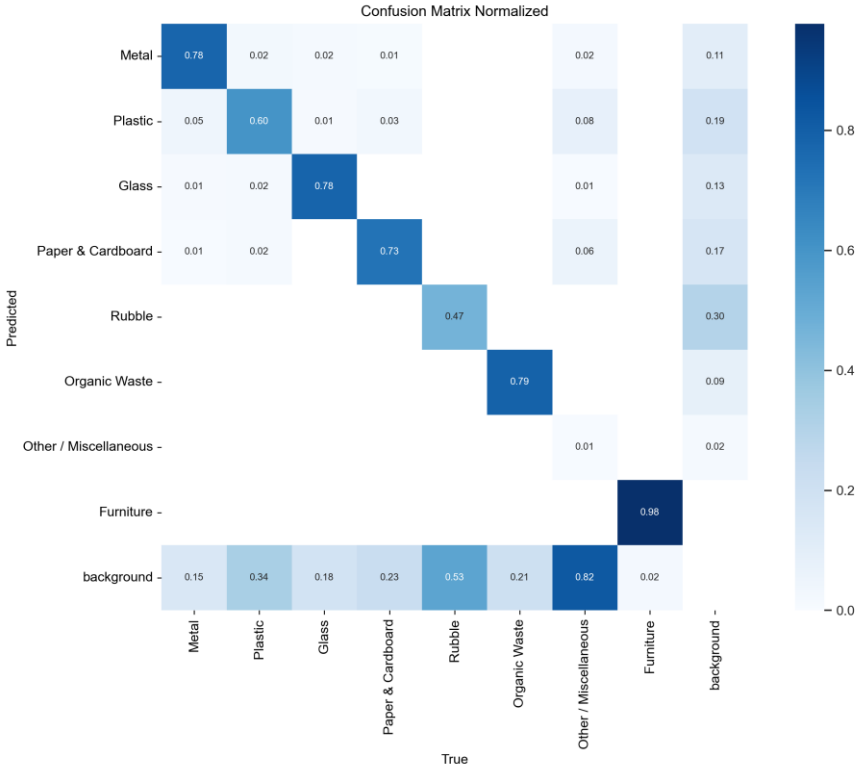
run	epochs	hyperparameters	mAP50-95	mAP50
train	50	batch=-1, imgsz=1024, optimizer=auto, lr0=0.001, lrf=0.01, momentum=0.95, weight_decay=0.0005, warmup_epochs=3	0.000	0.000
train1	15	batch=8, imgsz=640, optimizer=auto, lr0=0.001, lrf=0.01, momentum=0.95, weight_decay=0.0005, warmup_epochs=3	0.198	0.231
train2	250	batch=8, imgsz=640, optimizer=SGD, lr0=0.0001, lrf=0.01, momentum=0.95, weight_decay=0.0005, warmup_epochs=3	0.177	0.203
train3	250	batch=8, imgsz=832, optimizer=SGD, lr0=0.0001, lrf=0.01, momentum=0.95, weight_decay=0.0005, warmup_epochs=3	0.253	0.303
train4	100	batch=16, imgsz=416, optimizer=auto, lr0=0.001, lrf=0.01, momentum=0.95, weight_decay=0.0005, warmup_epochs=3	0.517	0.654
train5	60	batch=16, imgsz=416, optimizer=SGD, lr0=0.08962, lrf=0.01, momentum=0.6916, weight_decay=0.000461, warmup_epochs=3	0.496	0.635
train6	20	batch=16, imgsz=416, optimizer=SGD, lr0=0.01209, lrf=0.01, momentum=0.951, weight_decay=0.0005766, warmup_epochs=3	0.403	0.554
train7	60	batch=16, imgsz=416, optimizer=SGD, lr0=0.0001457, lrf=0.01, momentum=0.829, weight_decay=0.000685, warmup_epochs=3	0.518	0.657
train8	20	batch=16, imgsz=416, optimizer=SGD, lr0=0.001951, lrf=0.01, momentum=0.9574, weight_decay=0.0001339, warmup_epochs=3	0.496	0.641
train9	20	batch=16, imgsz=416, optimizer=SGD, lr0=3.087e-05, lrf=0.01, momentum=0.78, weight_decay=0.0004703, warmup_epochs=3	0.396	0.532

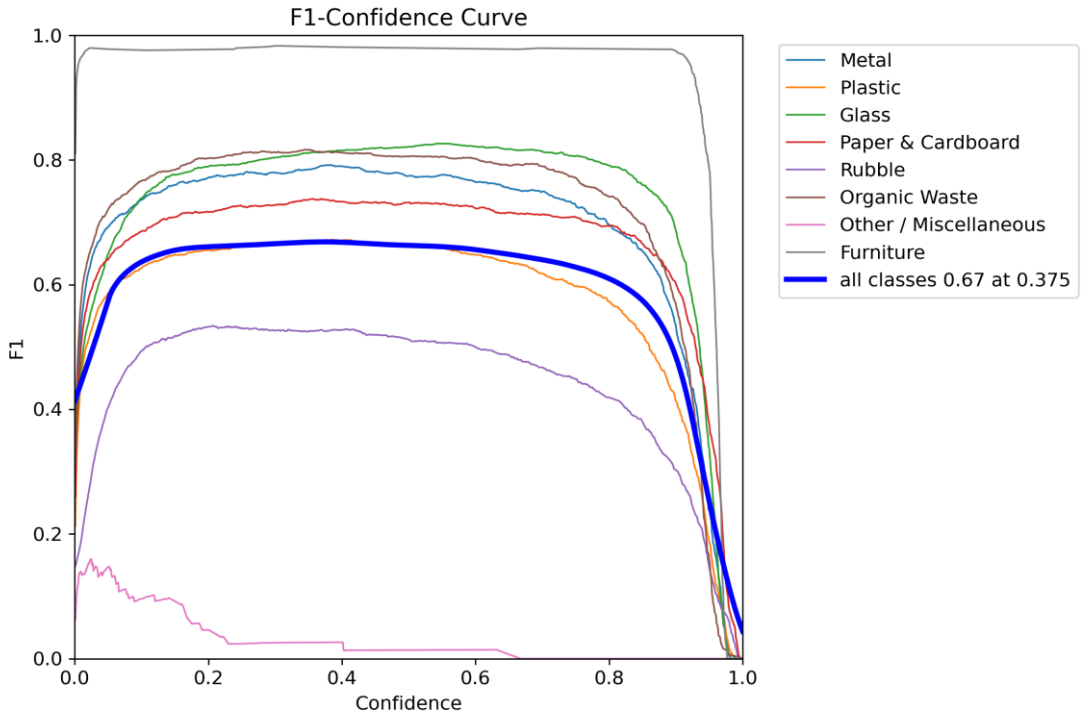
run	epochs	hyperparameters	mAP50-95	mAP50
train10	20	batch=16, imgsz=416, optimizer=SGD, lr0=0.007132, lrf=0.01, momentum=0.9382, weight_decay=0.0008566, warmup_epochs=3	0.441	0.591
train11	60	batch=16, imgsz=416, optimizer=SGD, lr0=2.529e-05, lrf=0.01, momentum=0.8691, weight_decay=0.0009723, warmup_epochs=3	0.455	0.599
train12	60	batch=16, imgsz=416, optimizer=SGD, lr0=0.00225, lrf=0.01, momentum=0.6407, weight_decay=0.0003717, warmup_epochs=3	0.545	0.677
train13	60	batch=16, imgsz=416, optimizer=SGD, lr0=0.0003358, lrf=0.01, momentum=0.7814, weight_decay=0.0001097, warmup_epochs=3	0.530	0.668
train14	60	batch=16, imgsz=416, optimizer=SGD, lr0=0.001826, lrf=0.01, momentum=0.6533, weight_decay=0.0005656, warmup_epochs=3	0.549	0.681
train15	60	batch=16, imgsz=416, optimizer=SGD, lr0=0.00121, lrf=0.01, momentum=0.6476, weight_decay=0.0008811, warmup_epochs=3	0.543	0.679
train16	20	batch=16, imgsz=416, optimizer=SGD, lr0=0.007745, lrf=0.01, momentum=0.8512, weight_decay=0.0008496, warmup_epochs=3	0.485	0.628
train17	60	batch=16, imgsz=416, optimizer=SGD, lr0=0.002268, lrf=0.01, momentum=0.8662, weight_decay=0.0003452, warmup_epochs=3	0.548	0.678
train18	20	batch=16, imgsz=416, optimizer=SGD, lr0=0.0209, lrf=0.01, momentum=0.8645, weight_decay=0.0007066, warmup_epochs=3	0.440	0.590
train19	60	batch=16, imgsz=416, optimizer=SGD, lr0=0.0001047, lrf=0.01, momentum=0.8908, weight_decay=5.13e-05, warmup_epochs=3	0.520	0.659
train20	20	batch=16, imgsz=416, optimizer=SGD, lr0=0.03768, lrf=0.01, momentum=0.8377, weight_decay=2.978e-05, warmup_epochs=3	0.434	0.585
train21	60	batch=16, imgsz=416, optimizer=SGD, lr0=0.02365, lrf=0.01, momentum=0.6052, weight_decay=3.647e-05, warmup_epochs=3	0.532	0.668

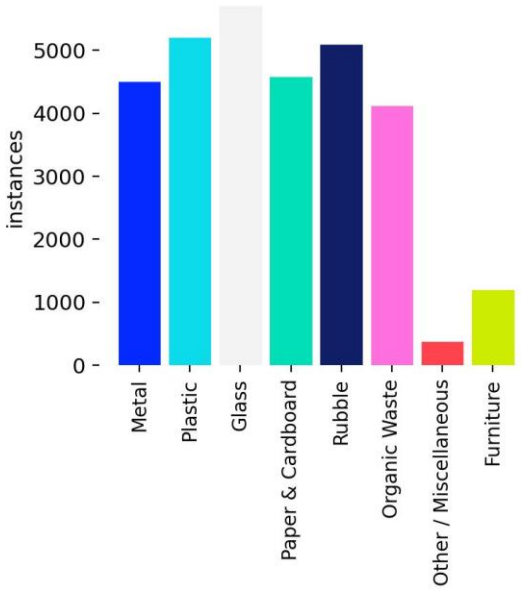
run	epochs	hyperparameters	mAP50-95	mAP50
train22	60	batch=16, imgsz=416, optimizer=SGD, lr0=0.0007302, lrf=0.01, momentum=0.9299, weight_decay=0.0005556, warmup_epochs=3	0.549	0.681
train23	20	batch=16, imgsz=416, optimizer=SGD, lr0=0.07207, lrf=0.01, momentum=0.7409, weight_decay=0.0008215, warmup_epochs=3	0.409	0.556
train24	20	batch=16, imgsz=416, optimizer=SGD, lr0=0.06377, lrf=0.01, momentum=0.8084, weight_decay=7.898e-06, warmup_epochs=3	0.416	0.564
train25	100	batch=32, imgsz=416, optimizer=SGD, lr0=0.0007302, lrf=0.01, momentum=0.9299, weight_decay=0.0005556, warmup_epochs=3	0.546	0.674
train26	50	batch=32, imgsz=416, optimizer=SGD, lr0=7.302e-05, lrf=0.01, momentum=0.9299, weight_decay=0.0005556, warmup_epochs=3	0.546	0.675

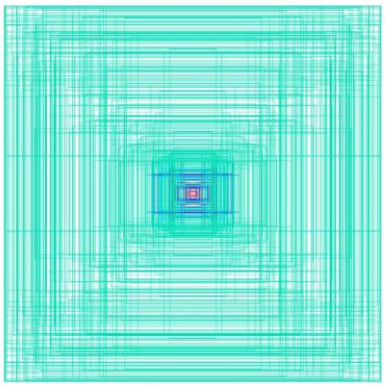
Gràfiques de mètriques del model de IA final

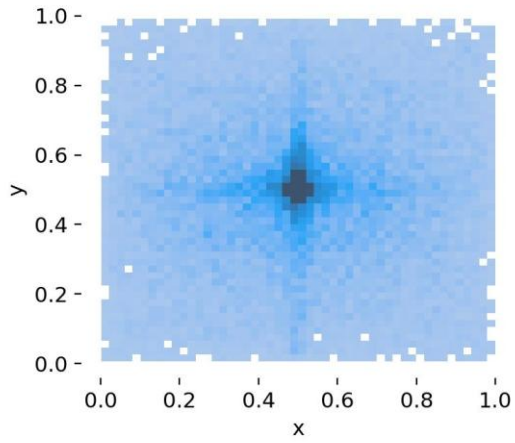


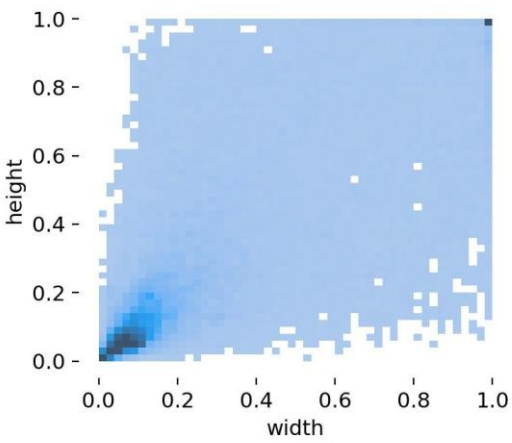


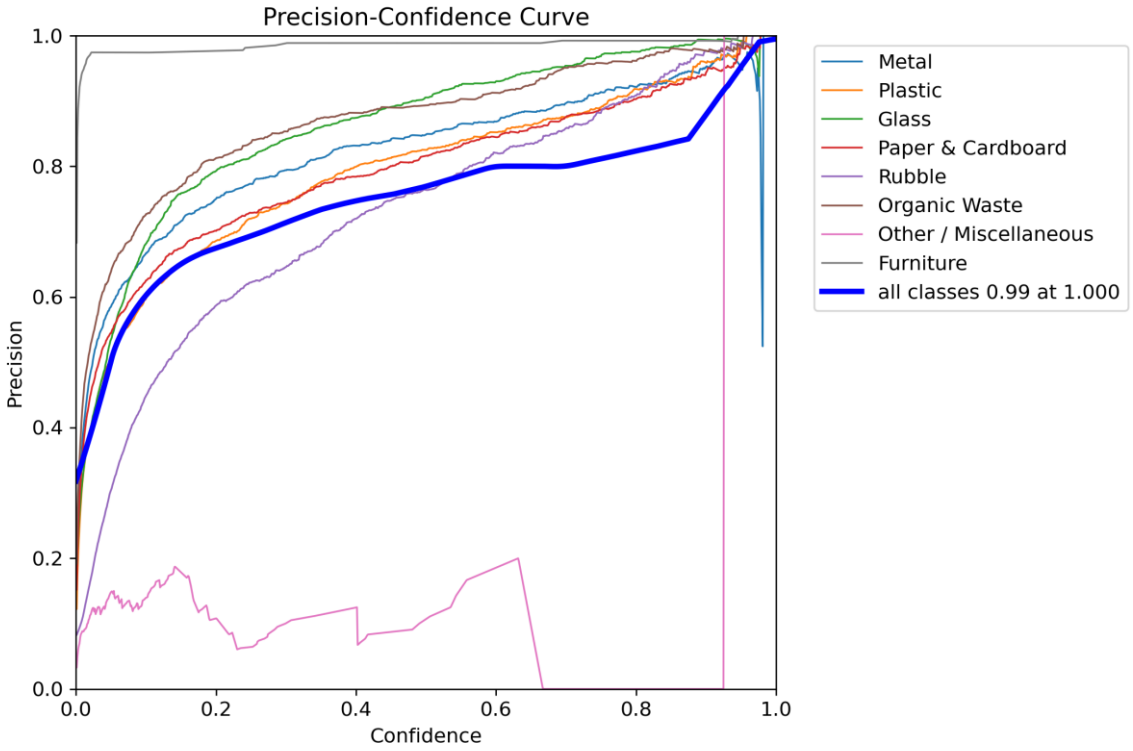

F1-Confidence Curve
F1
Confidence
Metal
Plastic
Glass
Paper & Cardboard
Rubble
Organic Waste
Other / Miscellaneous
Furniture
all classes 0.67 at 0.375

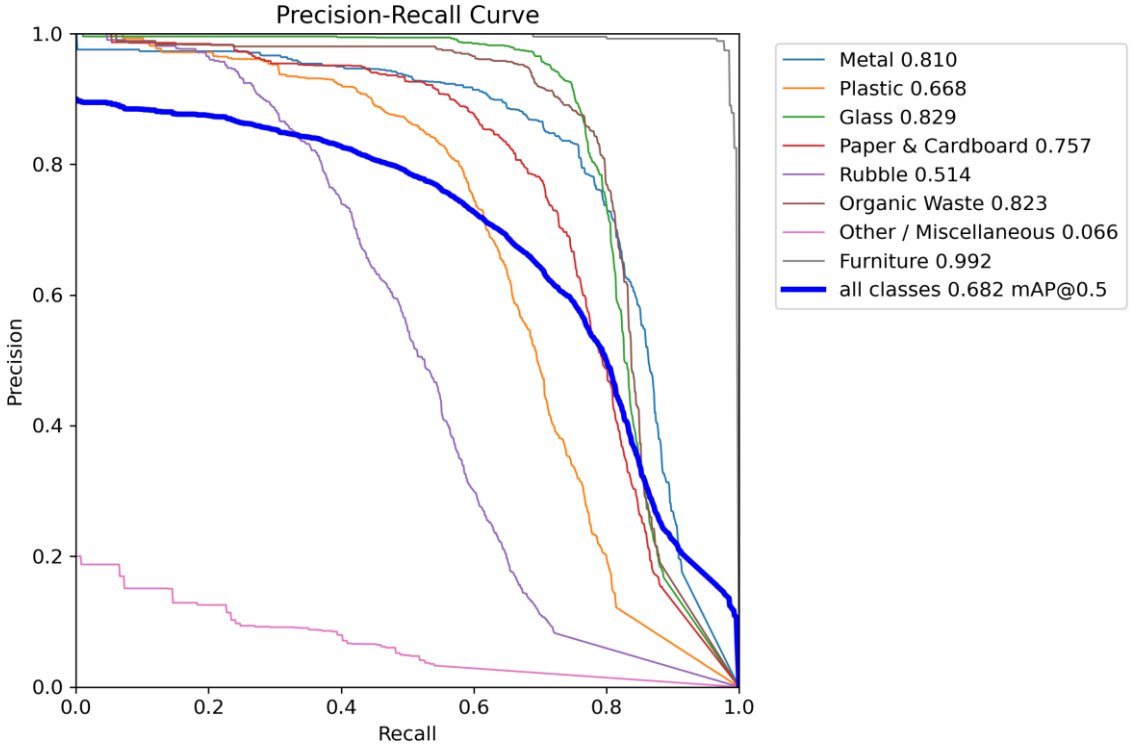

instances
Metal
Plastic
Glass
Paper & Cardboard
Rubble
Organic Waste
Other / Miscellaneous
Furniture

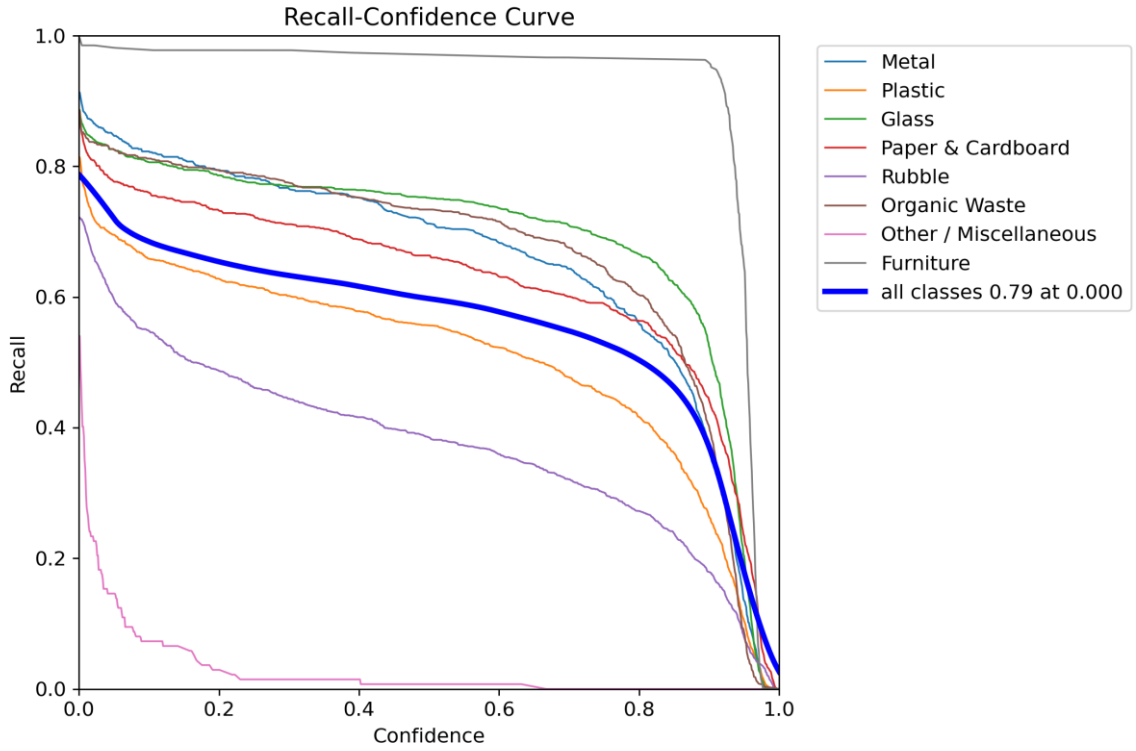


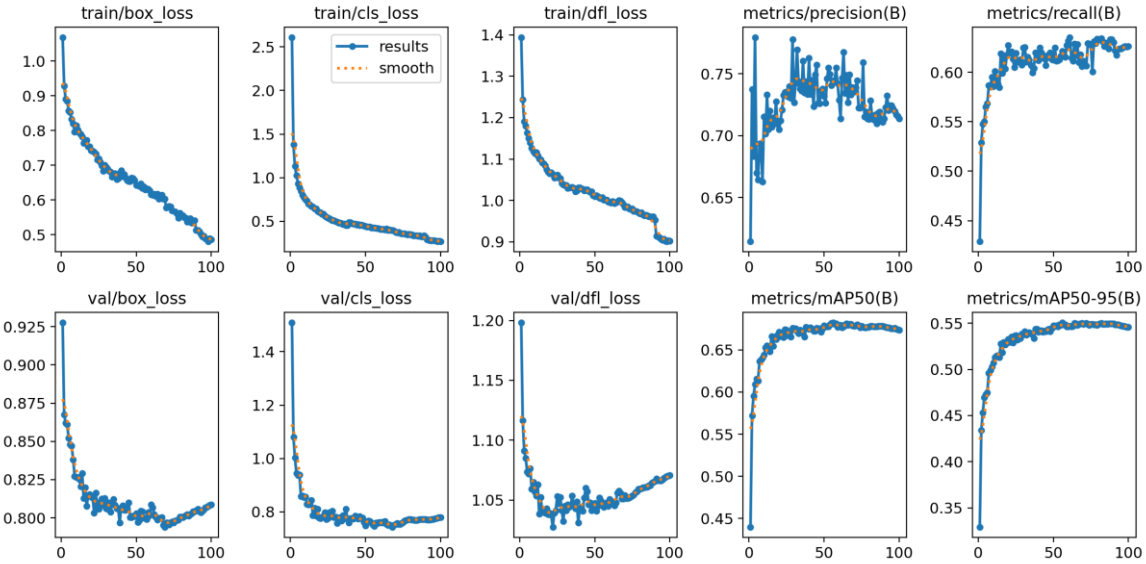

y
x


height
width


Precision-Confidence Curve
Precision
Confidence
0.0
0.2
0.4
0.6
0.8
1.0
Metal
Plastic
Glass
Paper & Cardboard
Rubble
Organic Waste
Other / Miscellaneous
Furniture
all classes 0.99 at 1.000


Precision-Recall Curve
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0
Metal 0.810
Plastic 0.668
Glass 0.829
Paper & Cardboard 0.757
Rubble 0.514
Organic Waste 0.823
Other / Miscellaneous 0.066
Furniture 0.992
all classes 0.682 mAP@0.5


Recall-Confidence Curve
Recall
Confidence
Metal
Plastic
Glass
Paper & Cardboard
Rubble
Organic Waste
Other / Miscellaneous
Furniture
all classes 0.79 at 0.000


train/box_loss
train/cls_loss
results
smooth
train/dfl_loss
metrics/precision(B)
metrics/recall(B)
val/box_loss
val/cls_loss
val/dfl_loss
metrics/mAP50(B)
metrics/mAP50-95(B)

Scripts

comarques_municipis.js – Script de mapeig de municipis amb les seves comarques

```
// This script generates a JSON file mapping comarques to their respective municipis
// It reads from a GeoJSON file containing municipis data and outputs a structured JSON file
const fs = require('fs')
const path = require('path')

const municipisPath = path.join(__dirname, '../data/municipis.json')
const raw = JSON.parse(fs.readFileSync(municipisPath, 'utf8'))

// L'estructura és GeoJSON: { type: 'FeatureCollection', features: [ ... ] }
const municipis = raw.features || []

if (!Array.isArray(municipis)) {
    throw new Error('No s\'ha trobat cap array de municipis al fitxer JSON (features[])')
}

const comarquesMap = {}

municipis.forEach(m => {
    const props = m.properties || {}
    const comarca = props.com
    const municipi = props.mun
    if (!comarca || !municipi) return
    if (!comarquesMap[comarca]) comarquesMap[comarca] = []
    comarquesMap[comarca].push(municipi)
})

fs.writeFileSync('comarques_municipis.json', JSON.stringify(comarquesMap, null, 4))
console.log('Fitxer comarques_municipis.json generat correctament!')
```

createDB.sql - Script d'Inicialització de Base de Dades

```
-- Crear taula "denouncer" (usuari)
CREATE TABLE denouncer (
    id SERIAL PRIMARY KEY,
    pfp TEXT, -- ruta del fitxer de la foto de perfil
```

```
    username VARCHAR(100),
    region VARCHAR(17),
    municipality VARCHAR(43),
    email VARCHAR(255) UNIQUE NOT NULL,
    paswd_hash TEXT NOT NULL,
    reset_token TEXT,
    reset_token_expiry TIMESTAMP WITH TIME ZONE,
    is_admin BOOLEAN DEFAULT FALSE
);

-- Crear taula "pending_denouncer" (usuari pendent de verificació)
CREATE TABLE pending_denouncer (
    id VARCHAR(64) PRIMARY KEY,             -- token de verificació
    email VARCHAR(255) UNIQUE NOT NULL,     -- correu de l'usuari
    paswd_hash VARCHAR(255) NOT NULL,       -- hash de la contrasenya
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP   -- moment de
creació
);


-- Crear taula "dumpster"
CREATE TABLE dumpster (
    id SERIAL PRIMARY KEY,
    name VARCHAR(11) NOT NULL UNIQUE,
    image TEXT NOT NULL, -- ruta del fitxer de la imatge
    longitude FLOAT NOT NULL,
    latitude FLOAT NOT NULL,
    region VARCHAR(17) NOT NULL,
    municipality VARCHAR(43) NOT NULL,
    date DATE NOT NULL,
    status VARCHAR(8) NOT NULL DEFAULT 'pending',
    description TEXT,
    denouncer INT NOT NULL,
    FOREIGN KEY (denouncer) REFERENCES denouncer(id)
);

-- Crear taula "rejected_dumpster"
CREATE TABLE rejected_dumpster (
    id INT PRIMARY KEY,
    date DATE NOT NULL,
    denouncer INT NOT NULL,
```

```
        FOREIGN KEY (denouncer) REFERENCES denouncer(id)
    );

    -- Crear taula "waste"
    CREATE TABLE waste (
        id SERIAL PRIMARY KEY,
        name VARCHAR(21) NOT NULL UNIQUE
    );

    -- Crear taula "dumpster_waste" per relació N:M
    CREATE TABLE dumpster_waste (
        dumpster_id INT NOT NULL,
        waste_id INT NOT NULL,
        PRIMARY KEY (dumpster_id, waste_id),
        FOREIGN KEY (dumpster_id) REFERENCES dumpster(id) ON DELETE
CASCADE,
        FOREIGN KEY (waste_id) REFERENCES waste(id) ON DELETE CASCADE
    );

    -- Inserir tipus de residu
    INSERT INTO waste (id, name) VALUES
        (0, 'Metal'),
        (1, 'Plastic'),
        (2, 'Glass'),
        (3, 'Paper & Cardboard'),
        (4, 'Rubble'),
        (5, 'Organic Waste'),
        (6, 'Other / Miscellaneous'),
        (7, 'Furniture');

    -- Assegurar-se de que el comptador seqüencial dels residus està a 7
    SELECT setval(pg_get_serial_sequence('waste', 'id'), (SELECT MAX(id)
FROM waste));
```

abocadorsbp.service – Script de Servei Systemd

```
    [Unit]
    Description=Uvicorn server for AbocadorsBP
    After=network.target

    [Service]
```

```
User=root
WorkingDirectory=/root/AbocadorsBPAPI
ExecStart=/root/AbocadorsBPAPI/.venv/bin/uvicorn app.main:app --host
0.0.0.0 --port 443 --ssl-keyfile
/etc/letsencrypt/live/abocadorsbp.com/privkey.pem --ssl-certfile
/etc/letsencrypt/live/abocadorsbp.com/fullchain.pem
Restart=always

[Install]
WantedBy=multi-user.target
```

Script de Compressió d'Imatges de la Inserció Inicial de Dades

```
from PIL import Image
import os

input_dir = "app/images/dumpsters"
quality = 70  # 0.7 * 100
max_size = (1024, 1024)

for filename in os.listdir(input_dir):
    if filename.lower().endswith((".jpg", ".jpeg", ".png")):
        path = os.path.join(input_dir, filename)
        try:
            with Image.open(path) as img:
                img = img.convert("RGB")
                img.thumbnail(max_size, Image.LANCZOS)
                img.save(path, "JPEG", quality=quality,
optimize=True)
                print(f"Comprimida: {filename}")
        except Exception as e:
            print(f"Error amb {filename}: {e}")
```