Disseny i implementació d'un xat en línia: OxApp

Daniel Hoyos Celdrán

4 de febrer de 2025

Resum– OxApp és un projecte de final de grau que consisteix en el disseny i implementació d'una aplicació web de missatgeria en línia completa, incloent-hi client i servidor. L'aplicació permet converses privades i grupals dins d'una aplicació web moderna i intuïtiva. Es construeix com una SPA amb React per al front-end, C# i .NET al back-end, i MongoDB com a base de dades, seguint una metodologia SCRUM per garantir un desenvolupament iteratiu i organitzat. S'han implementat mesures de seguretat com autenticació JWT i protecció contra atacs comuns.
Paraules clau– Missatgeria en línia, aplicació web, SPA, React, C#, .NET, MongoDB, autenticació JWT, seguretat web, SCRUM, grups, converses privades, xat en temps real, arquitectura client-servidor, WebSockets, API REST.

Abstract– Versió en anglès del resum OxApp is a final degree project focused on the design and implementation of a complete online messaging web application, including both client and server. The application enables private and group chats within a modern and intuitive web platform. It is built as a SPA using React for the front-end, C# and .NET for the back-end, and MongoDB as the database, following a SCRUM methodology to ensure iterative and organized development. Security measures such as JWT authentication and protection against common attacks have been implemented.
Keywords– Online messaging, web application, SPA, React, C#, .NET, MongoDB, JWT authentication, web security, SCRUM, groups, private chats, real-time chat, client-server architecture, WebSockets, REST API.

1 Introducció

Aquest treball de final de grau té com a objectiu dissenyar i desenvolupar, des de zero, un sistema complet de missatgeria en línia que inclogui tant el client com el servidor. El projecte neix de la motivació per aprofundir en el desenvolupament web full stack i en l'arquitectura d'aplicacions modernes dedicades a la comunicació.

L'aplicació, anomenada OxApp (acrònim de Online Xat Application), és una plataforma de missatgeria que busca ser reconeixible i senzilla. El nom combina la paraula catalana xat —equivalent a chat en anglès— amb una estructura similar a WhatsApp, deixant clar el propòsit de l'aplicació des del primer moment. En una versió més avançada, OxApp no només permetrà als usuaris mantenir converses privades o grupals, sinó que també introduirà una funcionalitat única: la possibilitat de proposar i compartir plans en grups d'amics o amb desconeguts. Aquesta característica diferencial pretén enriquir l'experiència social de l'usuari i dotar l'aplicació d'un valor afegit que la distingeixi de competidors com WhatsApp, Telegram o Discord. Tot i que encara no està implementada, és una de les propostes de millora que es faran més aviat.

• E-mail de contacte: danhoycel@gmail.com
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Valdimir Bellavista Parent (Department of Information and Communications Engineering)
• Curs 2024/25

Des del punt de vista tècnic, el projecte és una SPA (Single Page Application) que utilitza React com a llibreria principal per al desenvolupament del front-end, C# i .NET com a tecnologies per al back-end, i MongoDB com a base de dades. Aquesta selecció reflecteix un equilibri entre consolidar coneixements previs i adquirir habilitats en tecnologies modernes, amb l'objectiu de crear una solució robusta, escalable i eficient.

El desenvolupament del projecte segueix una metodologia SCRUM, dividint el procés en fases iteratives i incrementals per garantir un progrés constant i revisions contínues. Amb aquest enfocament, es pretén assolir un producte final funcional que no només compleixi els requisits plantejats, sinó que també serveixi com a base per al creixement personal i professional en els àmbits del desenvolupament web i la seguretat.

2 Motivacions

Tant les tecnologies i el desenvolupament web com les aplicacions de missatgeria sempre han estat un tema d'interès per a mi. Abans d'entrar al grau d'Enginyeria Informàtica ja programava, però arribar a desenvolupar una aplicació com aquesta semblava un repte molt complicat. És per això, aquest projecte em suposa un repte tan interessant. És una oportunitat per aprendre i millorar les meves habilitats de programació. Em motiva crear projectes com aquest i poder dedicar-hi el temps que durant el grau no he pogut.

3 Objectius

El projecte OxApp té com a finalitat principal el disseny i desenvolupament d'una aplicació amb la qual poder escriure's en temps real, que ofereixi una experiència funcional, segura i escalable per als usuaris. Amb aquest propòsit, s'han establert els següents objectius generals i específics:

3.1 Objectiu general

Crear una plataforma de missatgeria en línia que permeti als usuaris iniciar sessió, mantenir converses en temps real i interactuar amb altres persones, tot garantint una arquitectura sòlida i modular que serveixi com a base per a futures millores i ampliacions.

3.2 Objectius específics

- Desenvolupar un client web modern utilitzant React, que proporcioni una interfície intuïtiva i adaptativa per als usuaris.
- Implementar un servidor basat en C# i .NET que gestioni la comunicació en temps real mitjançant WebSockets i ofereixi una API REST per a operacions puntuals.
- Integrar l'ús de tokens JWT, per a gestionar les sessions dels usuaris.
- Proporcionar funcionalitats bàsiques, com l'enviament i recepció de missatges en temps real en converses privades i grupals.
- Establir mesures bàsiques de seguretat, com la protecció contra atacs comuns (XSS, Replay Attacks i Session Hijacking).
- Implementar una base de dades NoSQL (MongoDB) per emmagatzemar totes les dades.
- Fer ús de xifratge d'extrem a extrem (E2EE) per a totes les comunicacions.
- Afegir un factor social i diferenciador com és poder enviar propostes de plans a grups.

4 Estat de l'art

El desenvolupament d'aplicacions de missatgeria es troba en un moment d'evolució contínua, impulsat per la creixent demanda de solucions segures, ràpides i escalables. Aplicacions com WhatsApp, Telegram i Discord han establert un estàndard elevat pel que fa a funcionalitats i experiència d'usuari, incorporant tecnologies com el xifratge d'extrem a extrem, notificacions en temps real i sincronització multiplataforma. Aquestes aplicacions utilitzen arquitectures modernes que combinen serveis RESTful per a operacions síncrones amb WebSockets per a comunicacions en temps real, i algunes d'aquestes es recolzen en bases de dades altament escalables noSQL per gestionar milions de missatges de manera eficient.

En l'àmbit de seguretat, l'ús de JSON Web Tokens (JWT) s'ha consolidat com una pràctica estàndard per a la gestió d'autenticació, gràcies a la seva versatilitat i facilitat per treballar amb sistemes distribuïts. Paral·lelament, la gestió de sessions amb cookies HTTP only i mesures addicionals com la prevenció d'injeccions de codi han esdevingut essencials per garantir la integritat de les aplicacions. El desenvolupament d'interfícies d'usuari modernes, basades en frameworks com React, també ha facilitat la creació d'experiències d'usuari dinàmiques i accessibles, tot i que això, a vegades, implica un repte en termes d'optimització de rendiment i seguretat. En aquest context, el projecte busca combinar aquestes tecnologies i pràctiques per oferir una aplicació web de missatgeria funcional, segura i moderna.

5 Metodologia

Inicialment, el projecte es va plantejar seguint una metodologia en espiral per afavorir el desenvolupament iteratiu amb millores progressives. No obstant això, aquesta planificació mancava de precisió i realisme, fet que va requerir una reestructuració durant la primera fase del projecte.

Finalment, vaig adoptar una metodologia basada en SCRUM, dividint el projecte en diverses versions, cadascuna estructurada en sprints. Per garantir una millor organització, es va dur a terme una planificació a l'inici de cada sprint i una avaluació dels resultats al final. La figura 1 mostra la distribució de les versions, les dates i el nombre d'sprints de cada fase.

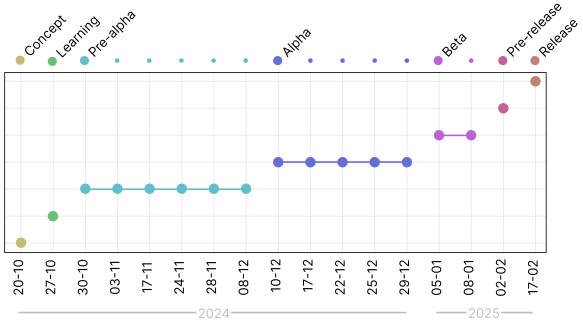

Fig. 1: Planificació del treball

Per a l'organització dels sprints i el seguiment de les incidències, inicialment vaig optar per Jira. No obstant això, després de dues setmanes de treballar amb l'eina, vaig veure, en un projecte desenvolupat únicament per mi, no era la millor opció.

Així, tota l'organització s'ha dut a terme mitjançant un calendari convencional i anotacions personals, un mètode que m'ha permès treballar de manera més flexible i centrar-me en les prioritats segons l'evolució del projecte. Per a cada versió, he establert al calendari la seva data límit i les dels seus respectius sprints. Durant cada sprint, he definit els objectius mitjançant una to-do list i he planificat la integració de les noves funcionalitats amb les ja existents i les previstes per al futur.

6 Tecnologies utilitzades

Per a poder dur a terme aquest projecte s'han utilitzat diverses tecnologies amb la intenció de tenir un desenvolupament eficient i una arquitectura robusta. Aquesta secció detalla les decisions preses quant a tecnologies usades per

al servidor, el client i la base de dades, així com les eines complementàries que han facilitat la feina i l'aprenentatge.

6.1 Servidor

Per al desenvolupament del servidor es van considerar diverses opcions, cadascuna amb característiques distintives que podien aportar diferents avantatges al projecte:

- **Servidor en C#** executat en entorn local amb la possibilitat de desplegament en un servidor extern.
- **Servidor en Python** sobre Docker, amb l'opció de desplegar el contenidor en entorns cloud.
- **Frameworks full-stack** com Remix o Next.js, que permeten integrar tant el front-end com el back-end en un únic entorn de desenvolupament.

Després d'una anàlisi comparativa, es va optar per la primera opció: **el desenvolupament del servidor en C#**. L'elecció va ser principalment motivada per la familiaritat prèvia amb aquest llenguatge i la seva integració amb .NET, que facilita la creació d'APIs robustes i segures. La decisió ha donat com a resultat una reducció de la corba d'aprenentatge i dedicar més temps a aprofundir en altres aspectes del projecte, com el desenvolupament del client.

6.2 Client

Pel que fa al client, es van avaluar diferents frameworks i llibreries front-end àmpliament utilitzat en la indústria, com React, Remix, Next.js, Angular i Vue. Tots aquests entorns presenten avantatges específics, però després d'una anàlisi exhaustiva, es va optar per **React** pels següents motius:

- **Versatilitat tecnològica:** React constitueix la base de molts altres frameworks populars, cosa que facilita l'aprenentatge de tecnologies relacionades en el futur.
- **Possibilitat d'ampliació:** Disposa d'un ampli conjunt de biblioteques i eines que permeten el desenvolupament tant d'aplicacions web, mòbils o d'escriptori mitjançant React Native. La qual cosa permetria exportar el projecte a entorns natius.
- **Adaptació a Single Page Applications (SPA):** React està optimitzat per a aplicacions d'una sola pàgina, millorant l'experiència d'usuari gràcies a la seva gestió eficient del DOM virtual.
- **Demanda en el mercat laboral:** L'alta demanda de React en l'àmbit professional incrementa la rellevància del projecte des d'una perspectiva de futur professional.

6.3 Base de dades

Per a la selecció del sistema de gestió de bases de dades (SGBD), es va dur a terme una anàlisi comparativa entre bases de dades relacionals (SQL) i no relacionals (NoSQL), considerant la naturalesa de l'aplicació i els requisits de flexibilitat i escalabilitat.

- **SQL (bases de dades relacionals):** Són ideals per a aplicacions amb dades fortament estructurades i relacions complexes, ja que organitzen la informació en taules amb esquemes predefinits utilitzant claus primàries i estrangeres.
- **NoSQL (bases de dades no relacionals):** Ofereixen una estructura de dades flexible sense la necessitat d'esquemes rígids. Són adequades per a aplicacions que requereixen escalar de manera eficient o gestionar dades semiestructurades.

Donada la necessitat d'escalabilitat i flexibilitat d'OxApp, es va optar per una base de dades NoSQL, concretament **MongoDB**. Aquesta elecció es justifica per diversos motius:

- **Model orientat a documents:** MongoDB utilitza documents en format JSON, facilitant la transició i la integració de dades entre el client i el servidor.
- **Suport oficial per a C#:** Proporciona una API molt complerta per a la integració amb aplicacions desenvolupades amb .NET.
- **Escalabilitat i rendiment:** Permet gestionar grans volums de dades amb un alt rendiment.

Aquesta elecció ha afavorit una integració senzilla amb la resta del sistema, assegurant una gestió eficient de les dades dins de l'aplicació.

6.4 Tecnologies complementàries

A més de les tecnologies principals, s'han utilitzat diverses eines que han contribuït a millorar la productivitat, l'organització i el disseny del projecte:

- **Figma:** Per al disseny de la interfície d'usuari i de tots els components de l'aplicació.
- **ChatGPT:** Com a suport durant l'aprenentatge de les noves tecnologies i com a referència ràpida d'algunes funcions com les de la llibreria de MongoDB.
- **Git i GitGub:** Sistema de control de versions per gestionar l'evolució del codi font de manera eficient.
- **Editors de codi (VS Code i Visual Studio 2022):** Eines principals per a la redacció de codi, depuració i gestió del projecte.

Aquest conjunt de tecnologies ha permès un desenvolupament àgil i estructurat, facilitant tant la fase de proves i aprenentatge com la de programació.

7 SEGURETAT

La seguretat és un aspecte fonamental del sistema, especialment en una aplicació web on es gestionen dades personals i les persones mantenen converses privades.

L'objectiu principal és protegir la confidencialitat, la integritat i la disponibilitat de la informació dels usuaris mitjançant l'aplicació de bones pràctiques i l'ús de tecnologies modernes. Els mecanismes de seguretat implementats són:

1. Autenticació basada en tokens (JWT).
2. Protecció contra atacs comuns.
3. Gestió segura de les credencials.
4. Auditoria i registres (logging).

A continuació es descriuen cadascun dels punts.

7.1 JWT

L'aplicació utilitza JSON Web Tokens (JWT) per autenticar els usuaris i mantenir sessions de manera segura. Es distingeixen dos tipus de tokens:

- **Access Tokens**: Tenen una validesa limitada i s'usen per accedir a recursos protegits.
- **Refresh Tokens**: Permeten generar nous access tokens quan aquests expiren, sense necessitat de reiniciar la sessió.

Els tokens d'accés s'envien de manera segura mitjançant cookies HTTP-only per evitar que el client accedeixi directament a ells. En canvi, els Refresh Tokens es guarden al localstorage del navegador i s'envien en la capçalera Authorization. Aquest últim token només s'envia en cas que el client rebi una resposta 401 Unautorized per part del servidor.

7.1.1 Funcionament

1. **Recepció de la consulta per part de l'API**
   - Abans d'identificar el recurs sol·licitat, es comprova si l'usuari té accés.
2. **Verificació de l'Access Token**
   - Si la cookie `"access-token"` existeix i és vàlida → *es concedeix l'accés*.
   - Si no és present o no és vàlida → *es continua amb la següent verificació*.
3. **Casos especials**
   - Consulta de "signin"o "signup"→ *es gestiona de manera específica*.
   - Sol·licitud de renovació de l'Access Token (recurs buit `HTTP /`) → *es verifica el Refresh Token*.
4. **Verificació del Refresh Token**
   - Si el `"refresh-token"` està present a la **capçalera Authorization** i és **vàlid** →
     - Retorna una *nova cookie* amb un *Access Token actualitzat*.
   - Si no és present o no és vàlid →
     - Retorna *401 Unauthorized*.

Aquest sistema assegura una **gestió segura i eficient** dels tokens, minimitzant riscos de seguretat i millorant l'experiència d'usuari.

A la figura 2 exposo dues situacions diferents en les quals es troba sovint el sistema i com en respon.

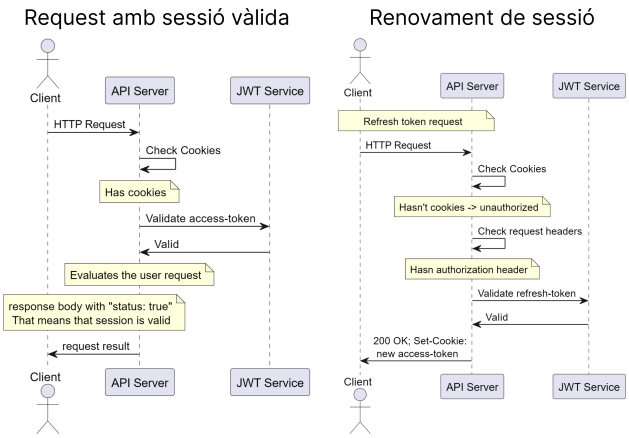

Fig. 2: Exemples de requests amb una sessió vàlida i de renovació de sessió.

7.2 Protecció contra atacs comuns

Les aplicacions web estan exposades a diversos atacs comuns que poden comprometre la seguretat dels usuaris i la integritat del sistema. Tot i que OxApp no està pensat per ser desplegat en producció, s'han implementat proteccions per a tres tipus d'atacs específics que són habituals en aquest tipus d'aplicacions:

1. Atacs Cross-Site Scripting (XSS)
2. Atacs de repetició (Replay Attacks)
3. Robatori de cookies (Session Hijacking)

Encara que existeixen altres tipus d'atacs, no s'ha considerat prioritari aprofundir-hi, ja que el focus principal d'aquest projecte és el desenvolupament d'una aplicació funcional, i no l'optimització completa de la seguretat. Per tant, les mesures de protecció implementades s'han centrat en riscos específics que poden ser mitigats amb tècniques conegudes i efectives.

Nota Actualment, el projecte no treballa amb TLS/SSL, ja que no es preveu el seu desplegament en producció i implementar-ho requeriria una inversió de temps significativa. Tanmateix, la incorporació de TLS/SSL seria una millora substancial per garantir la seguretat de les comunicacions.

7.2.1 Atac Cross-Site Scripting (XSS)

Els atacs de Coss-Site scripting consisteixen en injectar codi JavaScript maliciós a una web. Si s'aconsegueix es poden modificar dades i comportaments del lloc.

Per a evitar aquest tipus d'atacs, OxApp implementa proteccions al client i al servidor:

- **Al Client** s'evita l'ús de la funció ïnnerHTMLï utilitza funcions com çreateTextNode"que permeten evitar la interpretació d'HTML.

- **Al Servidor** s'escapen caràcters com: ≪", ≫", "&"... I es canvien per entitats HTML.

7.2.2 Atacs de repetició (Replay Attacks)

Aquest tipus d'atac consisteix a interceptar les dades que s'intercanvien dos nodes de la xarxa amb l'objectiu d'utilitzar-les de manera no lícita. En una aplicació com la meva, un atacant podria interceptar una de les consultes HTTP i raptar algun token i posteriorment obtenir accés a dades crítiques de l'usuari.

Per a acabar minimitzant o anul·lant les possibilitats d'un atac d'aquest tipus el que he fet és donar un temps de vida (TTL) curt als tokens d'accés. Concretament quinze minuts. A més, també hauré d'utilitzar identificadors únics (nonce) per a les comunicacions crítiques. Aquesta última protecció no està implementada encara.

7.2.3 Robatori de cookies (Session Hijacking)

El robatori de cookies és un atac que podem evitar fàcilment si utilitzem cookies HTTP only (fet), si marquem les cookies com a accessibles només per a HTTPS (actualment, en període de desenvolupament, no es farà) i si regenerem i validem el contingut delicat de les cookies periòdicament (fet amb el TTL curt dels access tokens).

7.3 Gestió de credencials

Les contrasenyes dels usuaris es protegeixen utilitzant xifratge amb l'algoritme bcrypt, que aplica un mecanisme de hashing adaptatiu. Aquest mètode és especialment resistent als atacs de força bruta, ja que incorpora un factor de cost que pot augmentar amb el temps per fer el càlcul més costós a mesura que augmenta la capacitat computacional. Això garanteix una protecció robusta per a les credencials, fins i tot davant possibles compromisos en el futur.

7.4 Auditories i logging

El sistema manté registres de tot el que passa com intents d'accés, errors i accions crítiques. Això permet detectar i respondre ràpidament a incidents de seguretat o qualsevol mena d'errors.

8 SERVIDOR

El servidor és el nucli del sistema, encarregat de gestionar la comunicació entre els usuaris i proporcionar una base robusta per al funcionament del xat. Actua com a intermediari per gestionar peticions, emmagatzemar dades i assegurar la seguretat i escalabilitat del servei.

L'arquitectura del servidor es divideix en tres blocs principals:

1. **API REST**: Servei encarregat de gestionar operacions com l'inici de sessió, registre d'usuaris, i altres interaccions estàtiques.

2. **Servidor de WebSockets**: Permet la comunicació bidireccional en temps real per a l'enviament i recepció de missatges entre els usuaris connectats.

3. **Connexió amb la base de dades**: Emmagatzema informació crítica com usuaris, missatges, i dades de configuració.

Ha estat desenvolupat amb C# i .NET, aprofitant la seva robustesa i el seu ecosistema per implementar funcionalitats complexes amb facilitat. A més, utilitza MongoDB com a base de dades NoSQL per gestionar dades no estructurades, adaptant-se perfectament a les necessitats d'un sistema de missatgeria. També incorpora mecanismes de seguretat com autenticació JWT i xifratge HTTPS per garantir la privadesa i seguretat de les comunicacions. Aquest enfocament modular facilita l'escalabilitat i permet afegir noves funcionalitats de manera eficient en el futur.

8.1 Flux d'execució

Per a poder tenir els dos servidors en marxa i compartir la informació, el servidor llença dos fils separats, un per a cada servei (API i servidor webSokets). Cadascun d'aquests treballa independentment i comparteixen l'accés a la base de dades que és gestionat per semàfors per a evitar problemes de seccions crítiques.

8.2 Estructura

Per donar servei a l'aplicació, el servidor es basa en tres blocs principals: una API, un servidor de WebSockets i el sistema de connexió amb la base de dades. Per garantir el funcionament eficient dels tres sistemes, el programa s'ha estructurat seguint el principi de responsabilitat única, assignant a cada secció una funció específica gestionada per actors dedicats. A la figura 3 es pot veure la relació entre els components de cada mòdul i com interactuen entre ells.

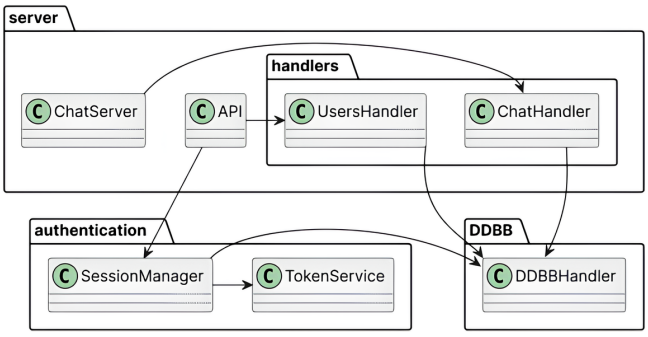

Fig. 3: Diagrama de classes del servidor

8.2.1 Classes auxiliars

A més de les classes principals, el servidor també compta amb algunes d'auxiliars i estructures que ajuden a tenir un millor codi:

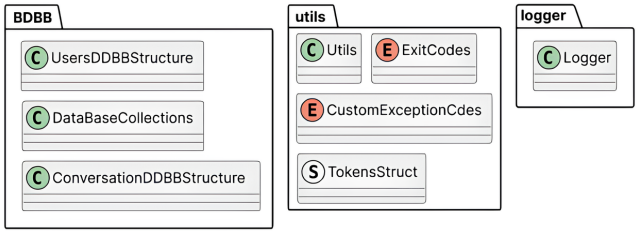

Fig. 4: Classes addicionals del servidor

8.3 API

El servidor proporciona una API encarregada de les peticions amb relació als usuaris, obtenció de dades i sobre les sessions. Està dissenyada per operar mitjançant peticions HTTP amb certa varietat de mètodes que faciliten la identificació del propòsit de cada petició de manera més eficient.

L'arquitectura modular permet ampliar funcionalitats de manera eficient, simplificant l'afegiment de nous endpoints i la reutilització de la lògica de processament. A més, pel que fa a seguretat estrictament de l'API, estableix una configuració CORS que regula l'accés entre orígens. Les respostes segueixen un format JSON estructurat per garantir una comunicació clara entre client i servidor.

8.3.1 Arquitectura

L'API està basada en peticions HTTP i suporta mètodes com `GET`, `POST`, `PUT`, `PATCH` i `OPTIONS`.

L'arquitectura és modular i això facilita l'addició de nous endpoints i funcionalitats. Per exemple:

- Afegir un endpoint nou només requereix una funció addicional a `HandleRoute`.
- La lògica de validació i processament es pot reaprofitar fàcilment mitjançant classes com `SessionHandler` i `UsersHandler`.

8.3.2 Recursos de la API

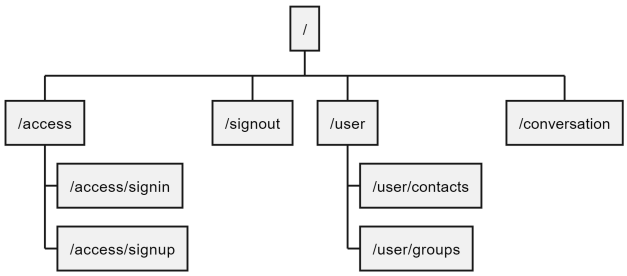

Fig. 5: Arbre de recursos de l'API

Cada recurs pot suportar diferents mètodes HTTP, cadascun amb una funció específica dins del sistema. A continuació, es detallen les operacions disponibles per a cada mètode:

- **GET**: Es fa servir per a recuperar informació de manera segura sense modificar l'estat del servidor. Aquest mètode és utilitzat per recuperar dades d'usuaris, com ara informació de perfil o llistes de contactes i missatges.
- **POST**: Es fa servir per a la creació de nous recursos o per a processos d'autenticació. Per exemple, aquest mètode és clau per registrar nous usuaris o iniciar sessions al sistema.
- **PUT**: Dissenyat per actualitzar o substituir completament un recurs existent. Un ús típic és gestionar operacions com *signout*, que requereix actualitzar l'estat de sessió de l'usuari.
- **PATCH**: Orientat a la modificació parcial d'un recurs ja existent. És útil per gestionar operacions específiques, com ara canviar l'estat d'un missatge (p. ex., llegit o no llegit).

Aquest model estructurat facilita una gestió clara i eficient dels recursos, seguint els estàndards de les API REST, i permet ampliar les funcionalitats de manera modular i escalable.

8.3.3 Gestió de Respostes

Per garantir una comunicació clara amb el client, totes les respostes de l'API es generen en format JSON. Això ens facilita l'enviament d'informació molt ben estructurada.

Totes les respostes contenen l'estat actual de la sessió, i el contingut o missatge corresponent.

Un exemple de resposta a una petició de contactes i amb la sessió activa té la següent forma:

```
{
        "status": true,
        "message": "...",
        "content": [<contactes>]
}
```

Respecte a l'estat de la mateixa resposta HTTP, es poden donar diferents casos: `BAD REQUEST 400`, `UNAUTHORIZED 401`, `INTERNAL SERVER ERROR 500`, `NOT FOUNT 404`, o bé `OK 200`.

8.3.4 Capacitats de Seguretat

L'API incorpora mesures de seguretat per protegir la integritat i confidencialitat de les dades. Una és la gestió d'autenticació ja explicada en detall a l'apartat 7.1 del capítol de seguretat (7.1), i l'altra és la configuració de Cross-Origin Resource Sharing (CORS). Que autoritza configuracions flexibles per a peticions de diferents orígens (Cross-Origin Resource Sharing):

- Permet mètodes com `GET`, `POST`, `PUT` , `PATCH`i `OPTIONS`.
- Habilita el suport per `Authorization Headers`.
- Admet credencials en cookies HTTP-Only.

Tanmateix, s'asseguren les cookies d'accés, que es configuren amb atributs `HttpOnly` i `Secure`, garantint que només siguin accessibles pel servidor i estiguin protegides durant la transmissió.

8.4 Servidor de missatgeria (web sockets)

El servidor de missatgeria és l'encarregat de gestionar la recepció, el reenviament i les confirmacions dels missatges. Està implementat com un servidor de WebSockets per a mantenir connexions persistents amb els clients i garantir una comunicació bidireccional en temps real.

8.4.1 Flux d'execució

Comença inicialitzant un *HttpListener* que escolta les connexions entrants al port `5000`. Quan es rep una sol·licitud, es verifica si es tracta d'una petició de WebSocket i en cas que ho sigui, es continua processant en un fil separat, assegurant l'acceptació de noves connexions.

Cada fil assignat a una connexió WebSocket:

1. Assigna el *WebSocket* rebut a un usuari identificat pel seu *username*.
2. Manté la connexió activa mentre el client no tanqui la sessió.
3. Gestiona la recepció de missatges per part del client, processant-los segons el tipus de missatge rebut.
4. Reenvia els missatges al destinatari corresponent o retorna una resposta d'error si el destinatari no està connectat.

8.4.2 Format dels missatges

El format dels missatges segueix l'estàndard JSON, ja que és lleuger, fàcil de "parsejarï ideal per a l'intercanvi de dades. Tots els tenen la següent forma:

```
{
        "type": "",
        "conversationId": "",
        "content": {}
}
```

Tipus de missatges:

Missatges d'usuari Aquests són els que contenen el mateix missatge que es vol enviar. El camp **`type`** dirà **`userMessage`** i el contingut seguirà la següent estructura:

- `from`: Usuari que envia el missatge.
- `to`: Usuari o grup destinatari.
- `type`: Indica el tipus de missatge (p. ex., `text` per a missatges de text).
- `read`: Flag per a saber si el missatge ha estat llegit o no i per qui.
- `content`: Contingut del missatge.

Exemple d'un missatge de text:

```
{
        "from": "user0",
        "to": "user1",
        "type": "text",
        "read": ["user1"],
        "content": "Hola, com va?"
}
```

La versió final del projecte no compta amb altres tipus de missatge que no siguin de text, però a l'apèndix 1.4 s'exposen algunes millores que implementaré més endavant en aquest camp.

Missatges de confirmació Per a aquest segon tipus, el camp de `type` conté la string `readConfirmation` i anirà acompanyat de la ID de la conversa i el nom d'usuari del qui l'ha llegit.

8.4.3 Recepció i reenviament dels missatges

Quan el servidor rep un missatge per part d'un client, s'encarrega de determinar el destinatari i de reenviar-lo.

El procés segueix els passos següents:

- **Identificació del destinatari**: Utilitzem els camps `to` i `type` presents al missatge JSON rebut per a identificar l'usuari o grup que l'ha de rebre.
- **Comprovacions de validesa**:
  - Es comprova si el destinatari existeix a la base de dades i al conjunt de connexions actives.
  - Es valida que l'emissor tingui a la llista de contactes al receptor.
- **Guardar a la base de dades**: Tots es guarden amb el flag `read` buit.
- **Reenviament del missatge**:

  També s'envien a l'emissor, per a assegurar que el missatge ha estat tractat correctament i per a simplificar el sistema.
  - Per als missatges d'una conversa privada, es busca la connexió associada al destinatari i reenvia el missatge utilitzant `SendAsync`.
  - Per a missatges dirigits a un grup, es localitzen totes les connexions actives dels membres del grup i reenvia el missatge a cadascuna d'ells.

8.4.4 Confirmació del missatges

Per garantir un seguiment precís de l'estat dels missatges, tots els missatges s'envien inicialment com a no llegits. Quan un destinatari rep un missatge i té el xat obert, s'envia automàticament una confirmació de lectura a través del canal WebSocket.

El sistema està dissenyat per gestionar correctament la confirmació de lectura tant per als usuaris que tenen la conversa oberta en el moment de rebre el missatge com per als que estan desconnectats o tenen el xat tancat. Quan un usuari obre una conversa amb missatges pendents de lectura, el servidor actualitza l'estat d'aquests missatges i els marca com a llegits, assegurant-ne la sincronització per a tots els participants.

8.5 Connexió amb la base de dades

Per a connectar amb la base de dades s'utilitza la llibreria oficial de C#: "MongoDB.Driver". Gràcies a aquesta llibreria es poden fer consultes fàcilment i ràpidament a la base de dades, aplicant filtres i fent seleccions concretes d'informació.

8.5.1 Accés a les dades

Per a poder accedir a tot a la informació, existeixen classes com ÜsersHandler", "SessionHAndlerï ÇhatHandler"que a través de "DDBBHandler"ens permeten tenir accés a tot.

9 CLIENT

El client representa la interfície principal per als usuaris del xat, i és la part visible del projecte. Aquest component s'encarrega de proporcionar una experiència d'usuari intuïtiva, visualment atractiva, i funcional; permetent als usuaris interactuar amb les funcionalitats del servidor.

L'arquitectura del client es basa en una SPA (Single Page Application) desenvolupada amb React.

La interfície d'usuari es divideix en tres pantalles principals:

- **Pantalla de presentació**: Dona la benvinguda als usuaris i els dirigeix a les accions principals, com registrar-se o iniciar sessió.
- **Pantalla d'inici de sessió i registre**: Permet als usuaris autenticar-se o crear un compte nou.
- **Pantalla del xat**: Espai principal on els usuaris poden enviar i rebre missatges, tant en converses privades com en grups.

El client inclou diverses funcionalitats destacades:

- **Interfície reactiva**: Adaptada per a diferents dispositius i resolucions, amb suport per a temes (clar i fosc).
- **Gestió de l'estat amb Context API**: Permet compartir informació com la sessió de l'usuari i la configuració del tema a través dels diferents components.
- **Connexió amb el servidor**: Mitjançant WebSockets per a la comunicació en temps real i l'API REST per a operacions puntuals.

El client també incorpora pràctiques modernes, com l'ús de CSS modular i transicions suaus per millorar l'experiència d'usuari. L'objectiu és garantir una experiència fluida, on els usuaris es puguin centrar en la interacció sense distraccions tècniques.

9.1 Estructura i arquitectura

L'arquitectura del client s'ha dissenyat seguint el patró **component-based architecture** de **React**, que divideix la interfície d'usuari en components reutilitzables i independents. Aquesta estructura modular facilita el desenvolupament, manteniment i escalabilitat del projecte.

9.1.1 Estructura dels components

Nota Entre parèntesis la referència a l'apèndix on es troben imatges de cada part.

- **Pàgines principals:**
  - **LandingPage (6):** La pantalla principal de l'aplicació la qual dona una mica d'informació sobre el producte i ofereix l'accés al xat.
  - **Xat (8):** La pantalla principal de l'aplicació (quan es té una sessió activa), que inclou els següents components:
    - **Contactes:** Mostra els d'usuaris i grups amb els quals l'usuari pot interactuar.
    - **Missatges:** Gestiona les bafarades on es mostren els missatges enviats i rebuts. Incloent el seu contingut i informació, com l'hora i contacte.
  - **Inici de sessió i registre (7):** Formularis per autenticar o registrar nous usuaris.
- **Components:**
  - **Buttons (9):** Botons presents a tota l'aplicació.
  - **Navbar:** Per navegar entre diferents seccions de l'aplicació.
  - **Temes:** Per canviar entre modes (clar i fosc).

9.1.2 Flux de dades

L'aplicació utilitza un patró de gestió d'estat centralitzat, basat en **React Context API** i **hooks**, per assegurar que els components comparteixin informació de manera eficient:

- **State Management:** Permet que informació com l'estat de la sessió, els missatges i els contactes estiguin accessibles des de qualsevol component.
- **Prop Drilling (minimitzat):** Encara que alguns components passen dades a subcomponents, l'ús de Context evita la necessitat de passar dades per moltes capes. Aquest context s'utilitza per a compartir la informació de la sessió i del tema.

9.1.3 Comunicació amb el servidor

Comunicació amb l'API REST La comunicació amb l'API REST s'utilitza per gestionar operacions com iniciar sessió, registrar usuaris o obtenir l'historial de missatges. Per part del client, amb cada petició procura identificar-se i en cas de rebre una resposta unauthorized, fa una petició de renovament de tokens. Si aquest resulta exitós: repeteix la comunicació inicial. En cas contrari: tanca la sessió.

Missatges (servidor WebSockets) El servidor de WebSockets permet una comunicació en temps real, establint una connexió bidireccional i persistent entre el client i el servidor. Aquesta connexió s'utilitza per gestionar l'enviament i la recepció immediata de missatges, així com per emetre notificacions en temps real sobre esdeveniments, com l'arribada de nous missatges.

9.1.4 Disseny

L'estructura d'OxApp ha estat completament dissenyada utilitzant com a inspiració productes com WhatsApp, Discord, ChatGPT... La idea ha estat obtenir una interfície moderna i familiar per als usuaris, de manera que el canvi entre plataformes sigui el més suau possible.

L'aplicació compta amb colors i transicions agradables, missatges per a l'usuari clars i poc agressius i en general una experiència d'ús satisfactòria.

9.1.5 Modularitat

L'ús de **CSS modules** o **styled-components** permet encapsular els estils de cada component, mantenint l'ordre i evitant conflictes. A més, l'arquitectura suporta un sistema de temes, cosa que permet transicions fluides entre mode clar i mode fosc.

Aquest disseny estructurat assegura una experiència d'usuari coherent, mentre que l'arquitectura modular facilita l'escalabilitat del projecte.

9.1.6 User Experience

Respecte l'experiència d'usuari, el projecte ha estat pensat per a ser intuïtiu i senzill. Amb les opcions limitades al màxim per a oferir una experiència despreocupada.

10 Base de Dades

La base de dades és l'element central d'emmagatzematge del sistema, encarregada de gestionar de manera eficient i segura tota la informació necessària per al funcionament de l'aplicació de xat. S'ha seleccionat MongoDB com a gestor de base de dades per les seves característiques de flexibilitat i escalabilitat, adequades per a aplicacions que treballen amb dades no estructurades o semiestructurades, com els missatges i les relacions entre usuaris.

10.1 Estructura

Per a organitzar la informació s'utilitzen col·leccions. Cada una d'aquestes conté fitxers en format JSON amb una estructura personalitzada. OxApp fa servir quatre: `Users`, `Groups`, `Conversations` i `Sessions`.

Nota: A l'apèndix 1.3 es pot veure com queden totes les següents estructures.

10.1.1 Users

La col·lecció dels usuaris, com bé diu el seu nom, conté tots els usuaris que es donen d'alta a la plataforma. Cada usuari té les dades bàsiques (nom, nom d'usuari i contrasenya) a més de tots els grups i contactes que té. Cada usuari també guarda l'identificador de les converses, si està en línia o no i l'últim cop que ha estat connectat.

10.1.2 Groups

Per als grups l'estructura és més senzilla que la dels usuaris. Tenim la ID del grup, el seu nom, participants i la ID de la conversa associada.

Format del "groupId"o ID de grup El format del "groupId"seguirà el següent format:

- #gId-xxxxxx

Les sis "x"són un codi hexadecimal de sis xifres. Aquest codi ens permetrà també definir un color per al grup.

10.1.3 Conversations

Les converses guarden la següent informació: Tipus, si es tracta d'una conversa en grup o en xat privat; participants i missatges. Els missatges es guarden tots en format: from, to, content.

10.1.4 Sessions

Per gestionar les sessions, emmagatzemem el nom d'usuari, l'accessToken i el "refresh token"associat a la base de dades. Això ens permet tenir un control centralitzat de les sessions obertes i de l'accés de cada usuari. També simplifica accions com la identificació d'usuaris a partir del seu access o refresh token, o també facilita el tancament de sessió; ja sigui perquè els usuaris tanquen sessió manualment o bé perquè el servidor decideix fer-ho.

11 Resultats

El desenvolupament d'OxApp m'ha permès assolir diversos objectius clau, tot i enfrontar-me a reptes tècnics que han limitat l'abast d'algunes funcionalitats previstes inicialment.

11.1 Assoliments principals

Entre els aspectes més destacats del desenvolupament, cal ressaltar:

- La implementació d'una arquitectura escalable que separa clarament les funcionalitats del servidor i del client.
- La integració de la comunicació client-servidor a través de l'API REST, que gestiona de manera eficient operacions com l'inici de sessió, el registre d'usuaris i la consulta de dades bàsiques.
- El desenvolupament d'un servidor de WebSockets que permet als usuaris mantenir converses en temps real, garantint una experiència fluida.
- L'establiment d'un sistema de seguretat inicial basat en tokens JWT, que assegura una gestió segura de les sessions dels usuaris.
- El disseny i la implementació d'una interfície d'usuari moderna i funcional, que ofereix una experiència intuïtiva i atractiva, adaptada a diferents dispositius.

Tots els resltats han estat provats en un entorn localhost i segur, on es demostra que el projecte ha assolit l'objectiu principal: **crear una aplicació funcional de xat en línia**. Les decisions tècniques preses han estat efectives per establir una base robusta i escalable que servirà com a fonament per a futures versions.

11.2 Limitacions dels resultats obtinguts

Tot i els èxits esmentats, algunes funcionalitats clau han quedat pendents o han estat implementades parcialment, afectant l'abast complet dels objectius inicials:

- **Xifratge d'extrem a extrem**: Tot i la implementació d'un sistema de seguretat inicial, no s'ha pogut completar aquesta funcionalitat, que és essencial per garantir la privacitat avançada de les comunicacions.
- **Funcionalitats socials**: L'enviament de sol·licituds d'amistat i la possibilitat de participar plenament en grups han quedat incompletes, limitant l'experiència d'interacció social prevista.
- **Gestió de plans en grups**: Aquesta funcionalitat distintiva, que havia de diferenciar OxApp de competidors com WhatsApp o Telegram, no s'ha implementat, afectant el valor diferencial de l'aplicació.

Aquestes mancances són degudes a diversos imprevistos durant el desenvolupament. Ha estat necessari dedicar més temps del previst a estudiar tecnologies com React i JWT per assegurar una implementació robusta i segura.

També s'ha invertit temps extra en la refactorització de codi que inicialment no estava plantejat de manera òptima. Per exemple, l'API REST va ser completament refactoritzada per adaptar-se als estàndards de qualitat, i components clau de React, com la pàgina del xat, van requerir una revisió profunda.

11.3 Perspectiva dels resultats

En el seu estat actual, OxApp compleix amb l'objectiu principal del projecte: proporcionar als usuaris accés a un xat en línia funcional, on poden enviar i rebre missatges de manera efectiva. Això demostra la viabilitat de l'arquitectura implementada i la capacitat del sistema per gestionar les operacions essencials d'una aplicació de missatgeria.

Tot i que algunes funcionalitats previstes encara no han estat completades, els resultats obtinguts fins ara són significatius i proporcionen una bona base per a l'evolució futura d'OxApp.

12 Conclusions

El desenvolupament d'OxApp ha estat una experiència enriquidora i un repte significatiu, tant des d'un punt de vista tècnic com organitzatiu.

Al llarg del procés, he consolidat coneixements en tecnologies modernes com React, API i JWT, alhora que he adoptat bones pràctiques de desenvolupament. Aquest enfocament no només m'ha facilitat el desenvolupament, sinó que també queda una base sòlida per al creixement i la millora futura del projecte.

Malgrat els èxits assolits, el projecte presenta certes limitacions i àrees de millora. Algunes funcionalitats previstes, com el xifratge d'extrem a extrem o la gestió completa de plans, no les he pogut implementar. Aquestes mancances es deuen principalment a la necessitat d'ajustar les previsions de temps i a la complexitat de garantir una implementació robusta en altres àrees prioritàries, com l'API REST i la seguretat bàsica.

El procés ha representat una oportunitat per enfrontar-se a reptes reals del desenvolupament web, des de la necessitat de refactoritzar components fins a l'optimització de fluxos de dades i l'adaptació a les dificultats tècniques que han anat sorgint. Aquest aprenentatge em serà de gran utilitat per a futurs projectes.

En definitiva, el desenvolupament d'OxApp ha estat una experiència altament enriquidora, que m'ha permès tant adquirir coneixements tècnics, com també millorar la capacitat d'organització i planificació del treball. Tot i que els resultats poden ser millorats en el futur, són significativament satisfactoris. Tots aquests aspectes em permeten considerar el projecte com un èxit destacat, del qual estic profundament satisfet i motivat per continuar millorant-lo.

13 Repositoris

El codi del projecte està dividit en dos repositoris. Un per al servidor i un altre per a la app de React.

13.1 Enllaços

- OxApp (App de React)
- OxServer (Servidor)

Referències

[1] Next.js, *Documentació oficial de React, accessible a: https://react.dev/reference/react*.

[2] ReactJS Wiki, accessible a: https://www.reactjs.wiki/.

[3] React Documentation, *Learning React Fundamentals*, accessible a: https://react.dev/learn.

[4] MDN Web Docs, accessible a: https://developer.mozilla.org/en-US/docs/Web.

[5] Halodoc, *JWT vs Session Authentication*, accessible a: https://blogs.halodoc.io/user-authentication-jwt-vs-session/.

[6] IETF, *RFC 2616 - Hypertext Transfer Protocol (HTTP/1.1)*, accessible a: https://datatracker.ietf.org/doc/html/rfc2616.

[7] Tooltyp, *Architecture of an Instant Messaging Service*, accessible a: https://www.tooltyp.com/arquitectura-de-un-servicio-de-mensajeria-instantanea-como-whatsapp/.

[8] JWT.io, *Introduction to JSON Web Tokens (JWT)*, accessible a: https://jwt.io/introduction.

[9] Medium, *JWT Authentication in C# and .NET Core*, accessible a: https://medium.com/@sajadshafi/jwt-authentication-in-c-net-core-7-web-api-b825b3aee11d.

[10] IBM Docs, *What are JSON Web Tokens (JWT)?*, accessible a: https://www.ibm.com/docs/es/cics-ts/6.x?topic=cics-json-web-token-jwt.

[11] Microsoft Docs, *Cookie Class in C#*, accessible a: https://learn.microsoft.com/en-us/dotnet/api/system.net.cookie?view=net-8.0.

APÈNDIX

1.2 UI

1.2.1 Pàgines

OxApp compta amb tres pantalles: Una landing page per als usuaris sense una sessió oberta, una pàgina de ”signiní ”sign-nup”, i finalment la pàgina principal; el xat:

En primer lloc, trobem la landing page, on van a parar tots l'usuari per primer cop.

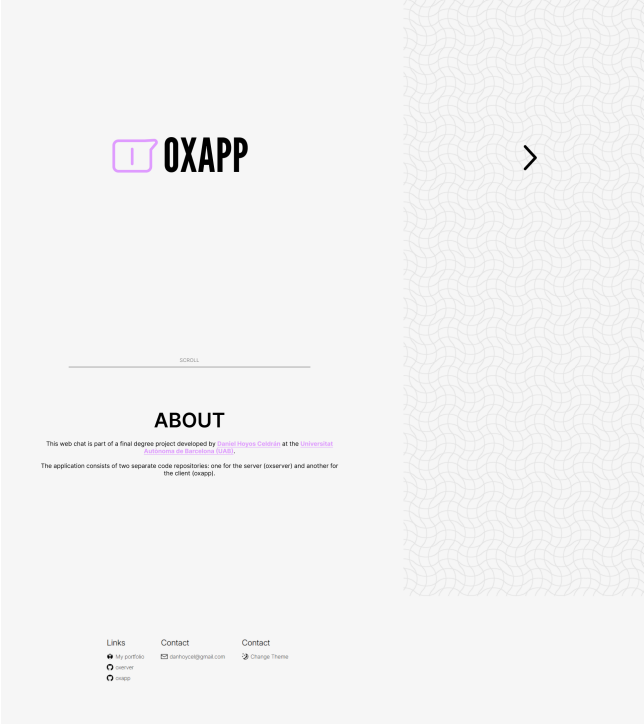

Fig. 6: Pàgina principal (landing page)

Seguidament, en accedir al xat es troba una pàgina on poder accedir al nostre compte o bé crear-ne un de nou.

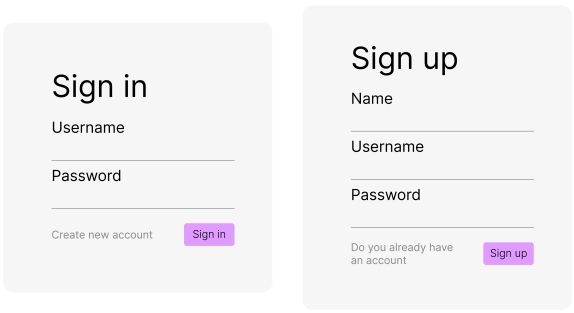

Fig. 7: Components d'accés a la plataforma

Finalment, la pantalla del xat. Aquesta està composta per un menú lateral on podrem accedir a les nostres converses i accions directes, com ara canviar el tema de l'aplicació o bé tancar sessió.

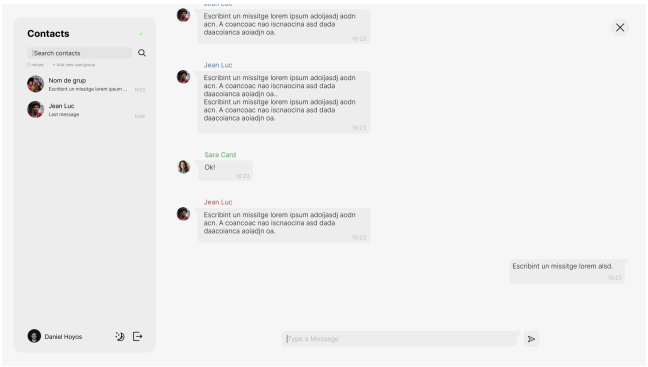

Fig. 8: Pàgina del chat

1.2.2 Components

L'aplicació compta amb tota una sèrie de components com ara botons, finestres emergents i més.

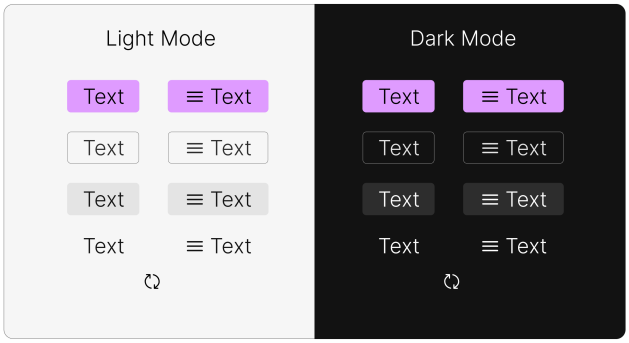

Fig. 9: Botons

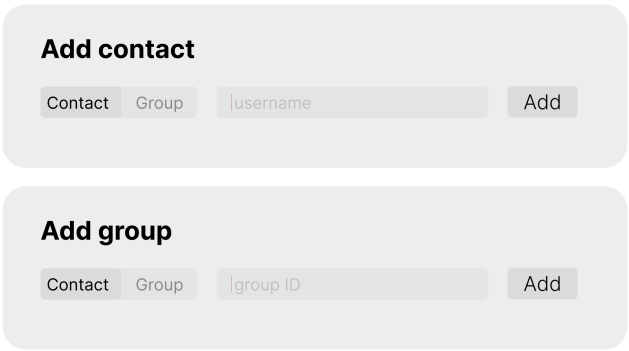

Fig. 10: Pantalla afegir grup o contacte

1.2.3 Colors

Després de valorar moltes opcions, finalment aquesta és la paleta de colors per a l'aplicació (11). La web està construïda sencera en escala de grisos: quatre per al mode fosc, quatre per al mode clar i un neutre que comparteixen.

El color d'accent i d'identitat de OxApp serà un fúcsia pastís (#DF9BFF).

A més, també s'han definit quatre colors base, amb els seus quatre respectius ”foreground”, per a notificacions i donar informació a l'usuari sobre èxits, errors...

Finalment, també s'han escollit tota una sèrie de colors per a donar personalitat als noms d'usuari dins de grups.

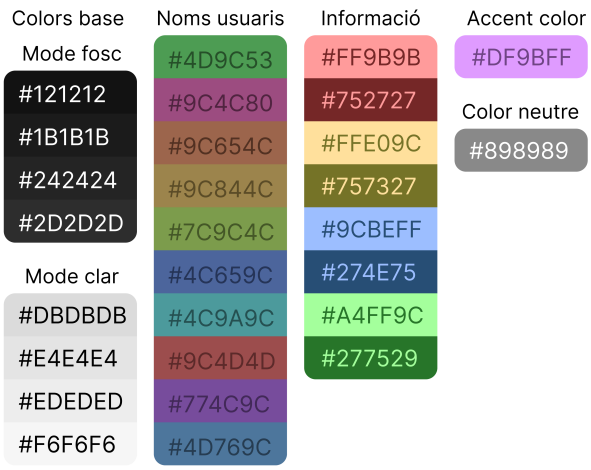

Fig. 11: Paletes de colors

1.3 Format documents Base de Dades

1.3.1 Users

```
        {
                "_id": {"$oid": ""},
                "username": "",
                "password": "",
                "name": "",
                "contacts": [
                {"name": "",
                        "username": "" },
                ...
                ],
                "groups": [
                {"name": "" ,
                        groupId": ""},
                ...
                ],
                "conversations": [
                "converationId": ""
                ],
                "lastSeen": <unix_time>,
                "isOnline": <bool>
        }
```

1.3.2 Groups

```
        {
                "_id": {
                        "$oid": ""
                },
                "groupId": "",
                "name": "",
                "participants": [...],
                "conversationId": ""
        }
```

1.3.3 Conversations

```
        {
                "_id": {
                        "$oid": ""
                },
                "type": "",
                "participants": [<usernames>],
                "messages": [
                {
                        "from": "",
                        "to": "",
                        "content": {
                                "type": "",
                                "content": ""
                        }
                },
                ...
                ]
        }
```

1.3.4 Sessions

```
        {
                "_id": {"$oid": ""},
                "username": "",
                "refreshToken": "",
                "accessToken": ""
        }
```

1.4 Futures millores

El projecte encara pot créixer molt i en molts aspectes. En aquesta secció veurem algunes de les possibles millores que es poden afegir tal com està el projecte.

1.4.1 Seguretat

- Els missatges es poden xifrar d'extrem a extrem.

1.4.2 Funcionalitats

- En comptes d'afegir contactes de manera directa: enviar sol·licituds d'amistat que s'hauran d'acceptar.
- Poder enviar plans als grups.
- Enviament de multimèdia.

Enviament de plans Aquesta és una funcionalitat que no hi ha hagut temps a implementar, però que afegeix un component diferenciador i social al xat. La idea és poder enviar formularis als grups d'amics o de desconeguts (com una comunitat). Aquest formulari quedaria ancorat al xat per a ser sempre visible fins a la data final del o fins que es cancel·li. A la figura 12 es pot veure la idea inicial del missatge.

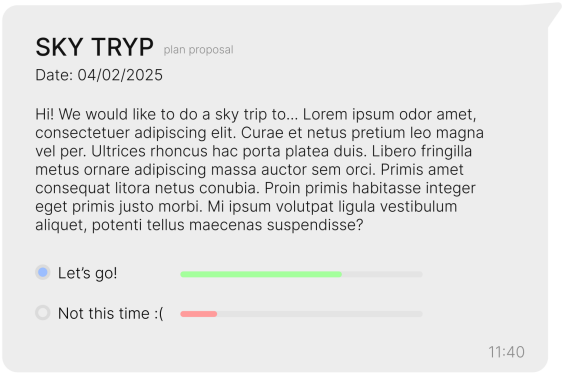

Fig. 12: Proposta de missatge per a un plà

ALTRES REFERÈNCIES

[1] FreeCodeCamp, *React Beginners Tutorial Part 1*, YouTube, accessible a: https://www.youtube.com/watch?v=jMy4pVZMyLM.

[2] FreeCodeCamp, *React Beginners Tutorial Part 2*, YouTube, accessible a: https://www.youtube.com/watch?v=7iobxzd_2wY&t=90s.

[3] Academind, *Implementing Sign-in in React*, YouTube, accessible a: https://www.youtube.com/watch?v=q4ywr3eZmk0&t=6074s.

[4] Jamie Burns, *Why You Should Avoid Using Result in Async Code*, Medium, accessible a: https://jamie-burns.medium.com/why-you-shouldnt-call-result-when-dealing-with-async-code-in-c-affee2142c86.

[5] Microsoft Docs, *ASP.NET HTTP Cookies Documentation*, accessible a: https://learn.microsoft.com/en-us/aspnet/web-api/overview/advanced/http-cookies.

[6] Academind, *Tutorial de gestió d'usuaris amb React*, YouTube, accessible a: https://www.youtube.com/watch?v=q4ywr3eZmk0.

[7] Mozilla Developer Network, *Writing a WebSocket Server in C#*, accessible a: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_server.