



ESCOLA TÈCNICA SUPERIOR  
D'ENGINYERIA  
Universitat Rovira i Virgili



# SISTEMES OBERTS

Curs 2024-25

PRACTICA 2

**Estudiants:** Oupman Miralles i Gerard Ros

**Professor/a:** Marc Sanchez Artigas

## **Índex**

Introducció .....	3
Estructura de la pràctica .....	4
Introducció.....	4
Organització de paquets.....	4
Decisions de disseny .....	15
Joc de proves .....	17
Conclusió .....	18
Manual d'instal·lació .....	18

## Introducció

En aquesta pràctica haurem de desenvolupar una aplicació web que permeti gestionar articles curts. L'objectiu és implementar un sistema que permeti publicar, consultar i gestionar articles. Aquesta aplicació es construirà utilitzant un back-end RESTful pels articles i un front-end funcional.

El sistema treballarà amb dos recursos principals:

- **Articles:** els articles podran tenir temes associats, ser públics o privats, i es podran ordenar per popularitat en funció del nombre de visualitzacions.
- **Usuaris:** els usuaris podran registrar-se i accedir als articles que estiguin disponibles segons el nivell d'autenticació i permisos.

Aquest projecte es desenvoluparà amb un enfocament escalable. A més d'integrar operacions bàsiques com la creació, lectura i eliminació d'articles, es contemplaran requisits específics com:

- Filtrar articles per autor o temes a través de consultes directes a la base de dades.
- Augmentar automàticament el nombre de visualitzacions quan es consulti un article.
- Gestionar la creació d'usuaris nous validant que la informació sigui correcta i coherent amb el sistema.

Aquesta aplicació també inclourà mecanismes de seguretat per protegir els recursos privats i garantir que només els usuaris autoritzats puguin fer algunes accions, com ara esborrar un article o veure'n el contingut complet si aquest és privat. Es treballarà seguint el model MVC, retornant les dades en format JSON.

En definitiva, aquesta pràctica no sol utilitzar conceptes com la integració de bases de dades i el desenvolupament de serveis RESTful, sinó que també planteja una arquitectura funcional que pot créixer i adaptar-se a escenaris reals.

# Estructura de la pràctica

## Introducció

La pràctica es basa en la creació d'un **sistema de publicació d'articles**, dissenyat amb una arquitectura moderna, escalable i segura. Per aconseguir una estructura clara i funcional, hem dividit el projecte en quatre blocs principals: **Controllers, Models, Services i Views**. Aquesta divisió segueix el patró **MVC (Model-View-Controller)**, facilitant la separació de responsabilitats, manteniment i escalabilitat del projecte.

## Organització de paquets

La nostra aplicació segueix una estructura modular, amb responsabilitats clarament definides per paquets. La idea era organitzar el treball de tal manera que fos fàcilment interpretable la funcionalitat de cada paquet i fos fàcilment escalable.

### [src/main/java/deim/urv/cat/homework2/controller](#)

Aquí tenim les classes que gestionen les peticions i respostes de la nostra aplicació web. Aquestes classes, conegudes com a controladors, són responsables de la lògica de negoci i de la interacció entre la vista (interfície d'usuari) i el model (dades).

#### 1) ArticleController

Aquest controlador s'encarrega de gestionar les peticions GET a la ruta "Article" i retorna la pàgina JSP corresponent per mostrar els articles. Utilitza injeccions de dependències per accedir als serveis i models necessaris per a la seva funcionalitat.

- **Característiques principals:**
  - **Path:** Article
  - Implementa el ArticleService
- **Mètodes:**
  - **showArticlePage:** s'encarrega d'agafar el id del article seleccionat i carregar la seva informació en la pàgina de article.
  - **searchArticles:** implementa la funcionalitat de la search bar. Vam afegir una funcionalitat al backend que a més de filtrar per topics i autor també filtres per text (contingut en el titol o body del article), llavors aquest metode busca els articles segons els paràmetres query, topics i autor i els retorna a la pàgina principal.

#### 2) AuthController

Aquest controlador s'encarrega de gestionar la autenticació dels usuaris.

- **Característiques principals:**
  - **UriRef:** Article
  - Implementa el AuthService
- **Mètodes:**
  - **showLoginPage:** s'encarrega de redirigir a la pàgina de log in.
  - **handleLogin:** a partir del usuari i contrassenya proporcionat en el form del login, al fer submit crida al authService i verifica si l'usuari està autenticat. En cas afirmatiu, agafa el context de sessió i es guarda el token del usuari, el username, el userId i la seva foto de perfil. En cas contrari redirigeix de nou a la pàgina de log in amb un missatge d'error.

### 3) HomeController

Aquest controlador s'encarrega de redirigir a la pàgina principal.

- **Característiques principals:**
  - **Path:** Home
  - Implementa el ArticleService
- **Mètodes:**
  - **showHomePage:** agafa tots els articles de la base de dades ordenats per visualitzacions de forma descendent, guarda en la sessió el nombre total d'articles i els envia a la pàgina principal per ser renderitzats.

### 4) LogoutController

Aquest controlador s'encarrega de borrar l'usuari de la sessió actual.

- **Característiques principals:**
  - **Path:** Logout
  - **UriRef:** logout
- **Mètodes:**
  - **handleLogout:** agafa el context de la sessió, invalida la sessió i redirigeix a la pàgina principal.

### 5) ProfileController

Aquest controlador s'encarrega agafar la informació del usuari de la sessió i mostrar la pàgina de perfil amb la seva informació.

- **Característiques principals:**
  - **Path:** Profile
  - Implementa el ArticleService
- **Mètodes:**
  - **showProfilePage:** agafa el id del usuari a partir de la sessió, i llavors busca el id del últim article publicat per aquell usuari. Si existeix busca el article i l'injecta a la pagina del perfil.

## 6) RegisterController

Aquest controlador s'encarrega de gestionar les peticions POST al registrar nous usuaris.

- **Característiques principals:**
  - **Path:** Register
  - **UriRef:** register
  - Implementa el UserService
- **Mètodes:**
  - **handleRegister:** a partir del usuari i password proporcionats en el form del register, busca si l'usuari no existeixi en la base de dades. En cas afirmatiu posa un missatge de confirmació i redirigeix a la pàgina de login. En cas contrari envia un missatge d'error i retorna a la pàgina de registre.

## 7) UserController

Aquest controlador s'encarrega de gestionar les peticions GET a la ruta "customer" i retorna els resultats a la pàgina de usuaris.

- **Característiques principals:**
  - **Path:** Users
  - Implementa el UserService
- **Mètodes:**
  - **showUsersPage:** crida al UserService i agafa una llista de UserDTO i redirigeix a la pàgina d'usuaris amb la llista d'usuaris.

## src/main/java/deim/urv/cat/homework2/exception

Aquí tenim les classes que gestionen les excepcions personalitzades de la nostra aplicació web. Aquestes classes són responsables de manejar errors específics que poden ocórrer durant l'execució de l'aplicació.

## 1) PropertyException

Aquesta classe representa una excepció personalitzada que s'utilitza per gestionar errors relacionats amb propietats específiques de l'aplicació.

- **Característiques principals:**

- `propertyException(String message)`: Constructor que accepta un missatge d'error i el passa a la superclasse `RuntimeException`.

## src/main/java/deim/urv/cat/homework2/model

Aquí tenim les classes que representen les dades de la nostra aplicació web. Aquestes classes inclouen entitats JPA, DTO i altres models que encapsulen les dades que exposem al client a través de l'API.

## 1) AlertMessage

Aquesta classe representa un missatge d'alerta que es pot mostrar a l'usuari. Inclou diversos tipus de missatges (èxit, advertència, perill, informació) i permet afegir errors específics.

- **Característiques principals:**

- `Type(enumeration)`: Enum que defineix els tipus de missatges (success, warning, danger, info).
- `Text(String)`: Text del missatge.
- `Code(String)`: Codi del missatge.
- `Errors(Error)`: Llista d'errors associats al missatge.

- **Mètodes adicionals:**

- `notify(Type type, String text)`: Notifica un nou missatge.
- `addError(String field, String code, String message)`: Afegeix un error al missatge.

## 2) Article

Representa un article a la base de dades.

- **Característiques principals:**

- `authorUsername (String)`: Nom d'usuari de l'autor de l'article.
- `authorProfileURL (String)`: link de la imatge de perfil de l'autor.
- `Id (Long)`: identificador de l'article.
- `imageURL (String)`: URL de la imatge destacada de l'article.

- `isPrivate` (boolean): Indica si l'article és públic o privat.
- `publishedAt` (Date): Data de publicació de l'article.
- `summary` (String): Resum breu de l'article.
- `title` (String): Títol de l'article.
- `views` (Integer): Número de visualitzacions de l'article.
- `Body` (String): cos de l'article.
- `Topics` (List<String>): llista de topics de l'article.

- **Mètodes adicionals:**

- `toString()`: Retorna una representació en forma de cadena de l'article.

### 3) SignUpAttempts

Aquesta classe controla els intents de registre d'usuaris per evitar abusos.

- **Característiques principals:**

- `MAX_ATTEMPTS` (int): Nombre màxim d'intents de registre permesos.
- `number` (int): Nombre actual d'intents de registre.

- **Mètodes adicionals:**

- `increment()`: Incrementa el nombre d'intents.
- `reset()`: Reinicia el nombre d'intents.
- `hasExceededMaxAttempts()`: Comprova si s'ha superat el nombre màxim d'intents.

### 4) User

Representa un usuari a la base de dades.

- **Característiques principals:**

- `imageURL` (String): enllaç de la imatge de perfil de l'usuari.
- `links` (Map<String, String>): hiperenllaç a l'últim article publicat.
- `username` (String): nom de l'usuari.

### 5) UserDTO

Representa un usuari però sense la informació dels hiperenllaços ja que no son necessaris en certs àmbits.

- **Característiques principals:**

- `imageURL` (String): enllaç de la imatge de perfil de l'usuari.
- `username` (String): nom de l'usuari.

## src/main/java/deim/urv/cat/homework2/service

Aquí tenim les classes que defineixen els serveis de la nostra aplicació web. Aquests serveis són responsables de la lògica de negoci i de la interacció amb les dades.

### 1) ArticleService

Aquesta interfície defineix els mètodes per gestionar els articles.

- **Característiques principals:**

- **getAllArticles():** Retorna una llista de tots els articles.
- **getArticleById(Long id):** Retorna un article pel seu identificador.
- **getLastArticleId(Long userId):** retorna el id del ultim article publicat per l'usuari.
- **searchArticles(String query, List<String> topics, String author):** busca articles segons els filters aplicats

### 2) ArticleServiceImpl

Aquesta classe implementa la interfície ArticleService i proporciona la lògica per gestionar els articles.

- **Característiques principals:**

- **WebTarget webTarget:** Objectiu web per fer peticions HTTP.
- **Client client:** Client HTTP per fer peticions.
- **BASE\_URI:** URI base per a les peticions, /rest/api/v1/article.

- **Mètodes:**

- **getAllArticles():** Retorna una llista JSON de tots els articles fent una petició GET.
- **getArticleById(Long id):** agafa el token guardat com atribut de sessió, inclou el path id al webTarget i el header Authorization amb el token ja que és un mètode privat i fa realitza una petició GET. Retorna el article que correspongui al id passat per paràmetre.
- **getLastArticleId(Long userId):** canvia la ruta a /customer i li afegeix el path id amb la capçelera Authorization i el token de la sessio per accedir a la ruta protegida. Amb el JSON de resposta el parsejem amb JsonReader i JsonObject ja que el id del ultim article està dins d'un string del camp

article, que es troba dins del camp links. Retornem el id parsejat o null en cas que no existeixi el camp links.

### 3) AuthService

Aquesta classe gestiona els mètodes de autenticació.

- **Característiques principals:**
  - **Client client:** Client HTTP per fer peticions.
  - **BASE\_URI:** URI base per a les peticions, /rest/api/v1/auth.
- **Mètodes:**
  - **Login(String username, String password):** realitza una petició POST a la ruta de autenticació del backend. Si retorna codi 200, agafem els camps token, userId i userPorfileURL de la resposta i els guardem com atributs del AuthService. Retornem un boleà a true conforme el login ha sigut satisfactori.

### 4) UserService

Aquesta classe gestiona els mètodes relacionats amb els usuaris

- **Característiques principals:**
  - **Client client:** Client HTTP per fer peticions.
  - **BASE\_URI:** URI base per a les peticions, /rest/api/v1/customer.
- **Mètodes:**
  - **getUserById(Long id, String token):** afegeix el path id a la ruta amb el id per parametre, afageix la capçalera de Authorization i retorna un JSON amb l'usuari trobat.
  - **getAllUsers(Long id):** retorna una llista de DTOs d'usuari amb tots els usuaris de la base de dades.
  - **register(String username, String password):** afegeix el path de register a la ruta, i fa un POST dels credencials de l'usuari sempre i quan es proporcionin els 2 camps. Retorna un boleà segons el codi de resposta del POST. Utilitza la classe RegisterRequest per tal d'enviar el username i password com a JSON.

 [src/main/webapp/WEB-INF/views/layout](#)

Aquí tenim els fitxers JSP que defineixen les diferents parts de la interfície d'usuari de la nostra aplicació web. Aquests fitxers s'utilitzen per construir la vista de l'aplicació. Ens hem basat en la coneguda web de Reddit.

#### 1) alert

Aquest fitxer gestiona la visualització dels missatges d'alerta a l'usuari. Utilitza JSTL per comprovar si hi ha missatges d'alerta o errors i els mostra en conseqüència.

- **Característiques principals:**

- flashMessage: Mostra un missatge d'alerta si n'hi ha.
- errors: Mostra els errors de validació si n'hi ha.

#### 2) articleCard

Aquest fitxer defineix la plantilla per mostrar una targeta d'article. Inclou informació com l'autor, la data de publicació, el títol, el resum, la imatge destacada, el nombre de visualitzacions i la privacitat de l'article.

- **Característiques principals:**

- authorUsername: Nom d'usuari de l'autor de l'article.
- publishedAt: Data de publicació de l'article.
- title: Títol de l'article.
- summary: Resum de l'article.
- imageURL: URL de la imatge destacada de l'article.
- views: Nombre de visualitzacions de l'article.
- isPrivate: Indica si l'article és privat.

#### 3) articleFig2

Aquest fitxer és practicament idèntic al articleCard però per tal de mostrar els articles en vista completa com la figura d'exemple 2.

- **Característiques principals:**

- Inclou les característiques de articleCard, exceptuant el summary
- Topics: topics de l'article
- Body: cos de l'article

#### 4) footer

Aquest fitxer defineix el peu de pàgina de l'aplicació web. Inclou informació de copyright i enllaços a la política de privacitat i els termes de servei.

- **Característiques principals:**

- Copyright: Mostra la informació de copyright.
- Links: Enllaços a la política de privacitat i els termes de servei.

## 5) navbar

Aquest fitxer defineix la barra de navegació de l'aplicació web. Inclou el logotip, la barra de cerca, els filtres de topics o autor i el botó de login.

- **Característiques principals:**

- Logo: Mostra el logotip de l'aplicació.
- Search Bar: Barra de cerca per buscar contingut a l'aplicació. Consumeix el input de text i l'envia al controlador de Articles per aplicar els filtres.
- Login Button: Botó per iniciar sessió. Aquest fa ús de scripts de JavaScript per tal de substituir-lo per la imatge i el nom de l'usuari de la sessió.
- Scripts i hrefs: tant per la search bar com el boto de login fem us de condicionals con when i otherwise i JavaScript per controlar el contingut que mostrem i a on redirigim l'usuari segons els accions que es realitzin.

## 6) sidebar

Aquest fitxer defineix la barra lateral de l'aplicació web. Inclou enllaços a la pàgina principal, la llista de usuaris, un article aleatori i la pàgina de Error404.

- **Característiques principals:**

- Home: redirigeix a la pàgina principal mostrant tots els articles per ordre de visualitzacions.
- Users: mostra una llista de tots els usuaris de la plataforma.
- Random Article: redirigeix a la pàgina d'un article seleccionat de forma aleatòria d'entre tots els articles del sistema.
- Error404: mostra la pàgina d'error per poder-la visualitzar i comprovar el seu funcionament.

## 7) userBanner

Aquest fitxer defineix les targetes que es mostren en la pàgina per mostrar el llsitat d'usuaris del sistema.

- **Característiques principals:**

- Imagte: mostra la imatge de perfil de l'usuari amb el camp imageURL.

- Paragraf: mostra el nom d'usuari amb del camp username.

## src/main/webapp/WEB-INF/views/pages

Aquí tenim els fitxers JSP que defineixen les pàgines principals de la nostra aplicació web. Aquests fitxers s'utilitzen per construir la vista de l'aplicació.

### 1) Article

Aquest fitxer defineix la pàgina per mostrar els detalls d'un article. Inclou la barra de navegació, la barra lateral i el peu de pàgina.

- **Característiques principals:**

- navbar.jsp: Inclou la barra de navegació.
- sidebar.jsp: Inclou la barra lateral.
- articleFig2.jsp: Inclou la tarjeta de l'article a mostrar com la Figura 2.
- footer.jsp: Inclou el peu de pàgina.

### 2) Home

Aquest fitxer defineix la pàgina principal de l'aplicació. Inclou la barra de navegació, la barra lateral, la secció principal amb els articles i el peu de pàgina.

- **Característiques principals:**

- navbar.jsp: Inclou la barra de navegació.
- sidebar.jsp: Inclou la barra lateral.
- footer.jsp: Inclou el peu de pàgina.
- Filter Section: Secció per filtrar els articles.
- articleCard.jsp: Inclou les targetes d'articles.

### 3) Login

Aquest fitxer defineix la pàgina d'inici de sessió a l'aplicació. S'encarrega d'agafar els camps de username i password i enviar-los al controlador.

- **Característiques principals:**

- Redirigeix a la uriRef login
- Permet anar a la pàgina de Register
- Consumeix els inputs username i password a través del boto de tipus submit dins del form.

## 4) Register

Aquest fitxer defineix la pàgina de registre a l'aplicació. S'encarrega d'agafar els camps de username i password i enviar-los al controlador per a realitzar el registre del nou usari.

- **Característiques principals:**

- Redirigeix a la uriRef register
- Permet anar a la pàgina de Login
- Consumeix els inputs username i password a través del boto de tipus submit dins del form.

## 5) Profile

Aquest fitxer defineix la pàgina de perfil de l'usuari de la sessió.

- **Característiques principals:**

- navbar.jsp: Inclou la barra de navegació.
- sidebar.jsp: Inclou la barra lateral.
- Main Section: consta de la foto de perfil i nom de l'usuari.
- articleCard.jsp: Inclou la targeta per mostrar l'últim article de l'usuari. En cas que no en tingui es mostra un text indicant-ho gràcies a les etiquetes <c: if>

## 6) Users

Aquest fitxer defineix la pàgina on es mostra el llistat d'usuaris de l'aplicació.

- **Característiques principals:**

- navbar.jsp: Inclou la barra de navegació.
- sidebar.jsp: Inclou la barra lateral.
- userBanner.jsp: Inclou les targetes dels usuaris amb un bucle forEach.
- footer.jsp: Inclou el peu de pàgina.

## Error404

Aquest fitxer no es troba en aquest paquet, però el mencionem ja que l'hem adaptat a l'estètica de la nostra pàgina. Es compon de un contenidor que conté un header indicant el 404, i un paràgraf que diu que la pàgina no s'ha pogut trobar. Finalment un botó que redirigeix a la pàgina d'inici cridant al controlador Home.

## src/main/webapp/resources/css

### 1) Tailwindcss

Nosaltres hem utilitzat la llibreria de Tailwindcss per tal de fer el css de l'aplicació. Per tal de poder-ho fer, hem inicialitzat un projecte de Node en el directori projecte/tailwindcss. A continuació hem instalat la llibreria de tailwind i inicialitzem la llibreria.

Això ens genera un fitxer tailwind.config.js el qual editem el seu camp `content[]` i li posem el directori on es troben les vistes (views) amb `**/*.jsp` per tal de que tailwind faci un seguiment de tots els arxius jsp per generar el css. Creem un nou fitxer que es digui input.css el qual hi incloem les directives `@tailwind base, component i utilities`.

Per fer que el css de les classes s'actualitzi en temps real, inicialitzem la eina de CLI amb la comanda `npx tailwind -i input.css -o tailwindcss.css --watch`, on el fitxer de sortida es el tailwindcss i el flag watch ens farà que s'actualitzi aquest fitxer al fer una modificació de classes en els jsp.

## Decisions de disseny

### 1. Arquitectura MVC

S'ha optat per implementar l'aplicació seguint el patró MVC tal i com se'ns demanava a l'enunciat de la pràctica:

- **Model:** Gestiona la lògica de dades i la persistència a la base de dades, utilitzant entitats JPA.
- **Vista:** Representa les pàgines JSP per a l'interfície d'usuari.
- **Controlador:** Classes que gestionen la lògica d'aplicació i la comunicació entre el Model i la Vista.

Aquest patró assegura una separació clara de responsabilitats, cosa que facilita el manteniment i l'escalabilitat del projecte.

### 2. Persistència amb JPA

S'ha utilitzat JPA (el que hem fet a la primera pràctica) per gestionar la persistència de dades a la base de dades relacional. Les entitats principals inclouen:

- **Article:** Representa els articles publicats amb camps com títol, resum, cos, imatge, visualitzacions i privacitat.

- **User:** Representa els usuaris registrats amb camps com nom d'usuari i contrasenya.
- **Topic:** Etiquetes o categories associades als articles.

### 3. Seguretat amb JWT

S'ha implementat un sistema d'autenticació basat en tokens JWT per garantir un accés segur als recursos privats de l'aplicació, fet també a la primera pràctica. Aquest enfocament ofereix:

- **Autenticació sense estat:** Els tokens s'envien amb cada petició, eliminant la necessitat de mantenir sessions al servidor.
- **Validació de tokens:** Cada petició a rutes protegides valida el token, assegurant que només els usuaris autoritzats puguin accedir-hi.

### 4. Interfície d'Usuari

La interfície s'ha dissenyat amb HTML i CSS (utilitzant TailwindCSS) per assegurar un disseny responsiu que s'adapta tant a dispositius mòbils com a escriptoris. Això inclou:

- **Llistat d'articles:** Amb filtres per autor i temàtica.
- **Pàgina web:** Per mostrar el contingut complet d'un article. Per aquesta ens hem basat en la famosa pàgina web coneguda 'Reddit'
- **Formulari d'autenticació:** Amb missatges d'error clars per a credencials invàlides.

### 5. Ampliacions Opcionals i Millors

- **Search Bar:** filtratge per text el qual busca coincidències en el títol i cos dels articles.
- **Register:** funcionalitat de registrar nous usuaris al sistema.
- **Backend:** modificacions i millors en alguns mètodes del backend per retornar millor la informació que volem mostrar en el client (com la cerca per text).
- **Logout:** funcionalitat per a poder desloguejar un usuari de la sessió.

Aquestes decisions de disseny han garantit un sistema modular, segur i escalable que compleix amb els requisits de la pràctica.

## Joc de proves

Per assegurar-nos que el projecte funciona correctament, hem realitzat diversos jocs de proves centrats en comprovar tant la funcionalitat de l'API com el de la pàgina web. El joc de proves s'ha basat en crear i desplegar la pàgina web amb la base de dades de la primera pràctica ja configurada, verificant que tot funciona com s'espera.

- **Proves de la pàgina web:**
  - Verificar que la pàgina principal mostra correctament els articles disponibles.
  - Comprovar que es filtra per autor i tòpics, assegurant que els resultats són els correctes.
  - Comprovar que a la pàgina d'un article es mostra tot el contingut associat, incloent el títol, resum, cos, imatge i informació de l'autor.
  - Validació de la responsivitat del disseny en diferents dispositius (mòbils, tauletes i escriptoris).
- **Proves de Seguretat:**
  - Comprovar l'autenticació amb credencials correctes, verificant que es genera un token JWT vàlid.
  - Verificar que l'accés a articles privats requereix un token JWT vàlid i que es retorna un error 401 per a usuaris no autènticats.
  - Proves d'inici de sessió amb credencials invàlides per comprovar els missatges d'error.
- **Proves de la Base de Dades:**
  - Comprovar que les dades surten correctament.
- **Proves d'Integració:**
  - Validació que els formularis d'inici de sessió funcionen correctament amb l'API.
  - Comprovació del correcte augment de visualitzacions en accedir a un article.

El joc de proves s'ha basat principalment en el BDD, centrant-se en el comportament esperat de l'aplicació des de la perspectiva de l'usuari final. Per a això, s'ha desplegat la pàgina web amb la base de dades configurada i s'ha comprovat que totes les funcionalitats visibles per a l'usuari funcionen correctament.

## Conclusió

En finalitzar aquest projecte, hem arribat a diverses conclusions i reflexions importants sobre el procés de desenvolupament i la seva aplicabilitat en el món real:

### 1. Aprendentatges clau:

- **Integració de tecnologies modernes:** Hem après a integrar un sistema back-end RESTful amb un front-end funcional, aplicant el model MVC per mantenir una arquitectura clara i escalable.
- **Gestió d'usuaris i autenticació:** Hem après a implementar sistemes d'autenticació i registre d'usuaris, assegurant que només els usuaris autoritzats tinguin accés a certes funcionalitats.
- **Treball amb bases de dades:** Hem guanyat experiència en la definició i gestió d'esquemes de bases de dades per donar suport a les operacions del sistema.

### 2. Aplicabilitat futura:

Aquest projecte és molt rellevant per a escenaris del món real. Les habilitats adquirides, com el desenvolupament d'API RESTful, la seguretat d'applicacions web i el disseny modular, seran útils en futures tasques professionals. Tot i potser no treballar amb les mateixes tecnologies, els fonaments teòrics i metodologies són pràcticament les mateixes, i formen la base per a molts projectes empresarials.

Llistat d'elements que més/menys ens han agradat :

Oupman	Gerard
Seguretat amb JWT	Creació pàgina web amb JSP
Interacció del model Vista/Controlador	Responsivitat
Desenvolupament de la API	Persistència mitjançant API REST
Desenvolupament del disseny	Tecnologies JAXB, JAX-RS, Java Persistence...
Limitacions / Restriccions de les Tecnologies de JPA, JAXB...	Seguretat amb JWT
IDE del Netbeans	Complexitat

## Manual d'instal·lació

1. Obrir projecte NetBeans.
2. Iniciar el server i connectar-se a la base de dades homework1.
3. Assegurar-se de la integritat del axriu install.jsp amb tota la informació a inserir a la base de dades.

4. Executar el projecte Homework1
5. Insertar la base de dades.
6. Executar el projecte Homework2

Nota: no hi hauria d'haver problema de css al executar l'aplicació. Pero si hi hagués cap error assegurar-se de que l'entorn de Node i la configuració de tailwind són correctes (Mirar organització de paquets/tailwind, pàg 15).