Software per a la planificació dels laboratoris de testing d'Applus+

Pol Pagès Luque

Resum—Applus+ és una empresa líder en el sector de certificació i assaigs. La seva divisió de Laboratoris, on he realitzat les meves pràctiques, afronta una necessitat creixent en relació amb la planificació efectiva dels seus laboratoris de *Testing, Inspection, and Certification (TIC)*. Aquest projecte busca solucionar els principals reptes que planteja la planificació dels laboratoris de test: desenvolupar una planificació visual i àgil, integrar les diferents eines corporatives per crear un ecosistema unificat, i oferir múltiples funcionalitats per a la generació d'informes de facturació, previsions de producció, planificacions de produccions mensuals, entre altres. L'objectiu del planificador de laboratoris és incrementar l'eficiència i eficàcia en la planificació de projectes i tasques, reduir els costos temporals, i millorar les relacions amb els clients actuals d'Applus, proporcionant confiança i flexibilitat.

Paraules clau—Applus+, CIMSA, CMB, Next.js, React.js, TypeScript, Front-End, Back-End, Azure, API, Prisma, Tailwind.

Abstract—Applus+ is a leading company in the certification and testing sector. Its Laboratories division, where I have completed my internship, faces an increasing need for effective planning of its Testing, Inspection, and Certification (TIC) laboratories. This project aims to address the main challenges posed by the test laboratories' planning: to develop a visual and agile planning process, integrate the various corporate tools to create a unified ecosystem, and offer multiple functionalities for generating billing reports, production forecasts, monthly production planning, among others. The goal of the laboratory planner is to increase efficiency and effectiveness in project and task planning, reduce time costs, and improve relationships with Applus' current clients, providing confidence and flexibility.

Index Terms—Applus+, CIMSA, CMB, Next.js, React.js, TypeScript, Front-End, Back-End, Azure, API, Prisma, Tailwind.

1 Introducció - Context del treball

La introducció s'ha tornat a redactar, i addicionalment s'ha afegit un exemple visual del que és la vista principal del planificador.

A l'era de la transformació digital, gestionar dades de manera eficient és crucial per a la rellevància operativa de les empreses. Aquest Treball de Fi de Grau té com a objectiu desenvolupar un sistema de planificació per a Applus+, líder en certificació i proves, que afronta desafiaments per la complexitat de les seves operacions i la diversitat dels seus projectes. La proposta és vital per millorar la coordinació i gestió de recursos als seus laboratoris TIC.

El sistema proposat, un planificador de laboratoris, integra eines corporatives en un ecosistema cohesiu i flexible, i permet una planificació visual i àgil. Això és crític per a tasques com, per exemple, proves de resistència en materials de construcció, on la coordinació d'equips i la precisió en la documentació són essencials per complir normatives internacionals.

El planificador utilitza Next.js i React.js per al *front-end*, oferint una interfície dinàmica, mentre que TypeScript i Prisma al *back-end* garanteixen la robustesa del sistema. A més, la plataforma facilita l'automatització de processos, com ara la gestió de la producció i la planificació paral·lela de tasques.

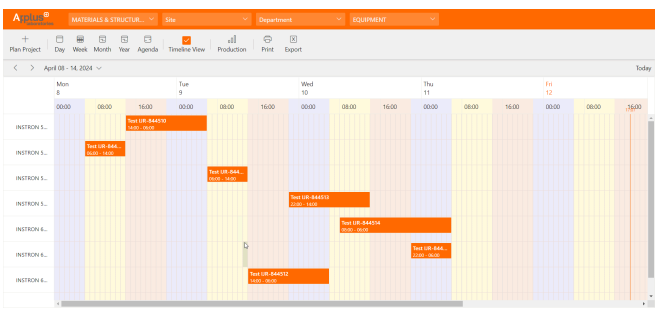


Fig. 1. Vista principal del CMB

En conclusió, el projecte busca augmentar l'eficiència i l'efectivitat en la gestió dels laboratoris d'Applus, reduint costos i enfortint relacions amb clients, establint una base per a futures millores operatives.

- *E-mail de contacte: polpages1999@gmail.com*
- *Menció realitzada: Enginyeria del Software*
- *Treball tutoritzat per: Marc Talló Sendra (Ciències de la Computació)*
- *Curs 2023/24*

2 Objectius

S'han corregit errors ortogràfics i sintàctics.
L'objectiu principal del projecte és desenvolupar un sistema de planificació per als laboratoris d'Applus que millori l'eficiència i l'eficàcia en la gestió de projectes i tasques.

A continuació s'exposen els objectius principals proposats a la fase de definició del projecte. Aquests objectius han estat definits a partir de les necessitats exposades pels directius dels laboratoris d'Applus:

1.- Integració de Dades
El primer objectiu del projecte és desenvolupar i implementar un sistema robust per a la integració de dades entre les diverses plataformes i sistemes utilitzats per Applus, com ara CIMSA (el *datalake* d'Applus Laboratories) i la base de dades del CMB (nom del planificador a Applus). Això inclou la creació i utilització d'APIs internes que facilitin la comunicació i la transferència de dades, assegurant que la informació del projecte estigui actualitzada i accessible. Aquesta integració ha de suportar la sincronització automàtica de tasques, recursos i projectes, eliminant redundàncies i millorant la precisió de la planificació.

2.- Interfície del Planificador
El segon objectiu es centra en el desenvolupament d'una interfície d'usuari intuïtiva i adaptable per al sistema de planificació. Aquesta interfície, basada en els principis de disseny de Fluent UI i utilitzant les últimes tecnologies de *front* i *back-end* per implementar-la, ha d'oferir als usuaris una experiència fluida i coherent.
Aquesta interfície ha de permetre una fàcil visualització i manipulació de projectes, tasques i assignacions de recursos a través de vistes personalitzables (dia, setmana, mes, any). L'accessibilitat i facilitat d'ús són prioritàries per garantir que tots els usuaris puguin gestionar eficientment els seus projectes.

3.- Funcionalitats de *Forecasting*
El tercer objectiu del projecte és implementar funcionalitats de forecasting que permetin generar informes de producció, així com la facturació i la gestió de recursos. Aquestes eines han d'aprofitar les dades integrades per oferir operacions precises que ajudin a la planificació estratègica i operativa. La capacitat d'anticipar necessitats de recursos i ajustar la planificació de projectes és fonamental per millorar l'eficiència i la satisfacció del client.

3 Estat de l'art

Aquesta secció no s'ha modificat al primer informe de progrés.
S'ha analitzat un ventall d'eines existents que ofereixen solucions de planificació. Hem recollit informació sobre plataformes com KendoReact Scheduler, DHTMLX Scheduler, FullCalendar i Syncfusion. També s'ha avaluat la possibilitat de construir des de zero el *software*, però els costos eren massa elevats en temps/persona.
S'han comparat les seves funcionalitats, interfícies d'usuari, integració amb altres sistemes, i adaptabilitat a les necessitats específiques de planificació de laboratoris de testing.

Tot i que totes les eines han presentat avantatges i inconvenients, Syncfusion ha sigut la que ha destacat per superar la majoria de les limitacions trobades en cada solució, enfocant-se en l'ecosistema Microsoft, la integració amb components de *front* i *back-end*, i la facilitació d'una experiència d'usuari més àgil i intuïtiva.

4 Metodologia

S'han modificat els estats possibles de cada tasca dins del Jira, ja que s'ha implementat una configuració per automatitzar el procés de monitorització de les tasques. També s'ha afegit una imatge de la configuració del taulell Kanban.
La metodologia aplicada en aquest projecte s'enfoca en *Kanban*. Aquesta metodologia àgil ens permet assegurar un procés de desenvolupament eficient i de qualitat.

Addicionalment, s'ha fet servir *git*, juntament amb *Azure DevOps*, per a mantenir un control de versions del codi font i facilitar el desplegament del programari.

4.1 Planificació i gestió del projecte

Per a la planificació i la gestió del projecte, s'utilitza Jira, una eina líder en la gestió de projectes de programari. Dins de Jira, s'ha configurat un tauler Kanban personalitzat per seguir el flux de treball del projecte amb els estats descrits més endavant. Aquest enfocament Kanban ens permet visualitzar el treball en curs en tot moment i gestionar el flux de tasques de manera eficient, permetent un lliurament continu.
Els estats proposats per a realitzar el seguiment de les tasques del projecte són els següents:

- **Assigned:** Tasques assignades a un membre de l'equip per desenvolupar-les.
- **In progress:** Tasques que estan en procés de desenvolupament per part del membre de l'equip assignat.
- **Pending info:** Tasques que necessiten de més informació per part d'algun dels interessats del projecte abans de poder continuar amb el desenvolupament.
- **Technical validation:** Tasques sotmeses a validació tècnica per part del responsable de l'equip i els desenvolupadors. Es valida que la funcionalitat s'ha implementat correctament
- **Functional validation:** Tasques sotmeses a validació funcional per part del responsable de l'equip i el responsable del departament corporatiu. Es valida que es compleixin els criteris d'acceptació establerts i les expectatives dels *stakeholders*.
- **Done:** Tasques que han estat completades satisfactòriament i estan a punt per ser integrades al projecte.

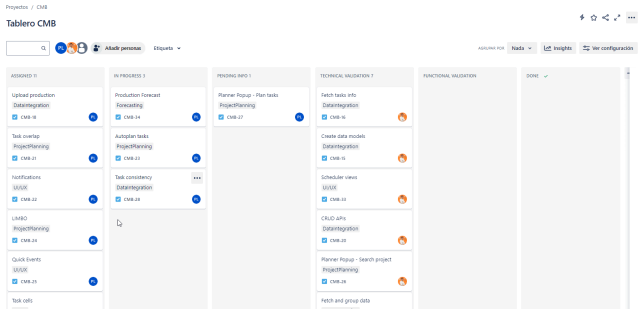


Fig. 2. Taulell Kanban Jira

Es pot trobar el taulell Kanban i la seva configuració a l'*Apèndix A2 i A3* respectivament.

4.2 Control de Versions

El control de versions del codi es realitza a través de *git*, un sistema de control de versions distribuït que facilita la col·laboració i gestió de canvis al codi font del projecte.

Per a la gestió de repositoris i branques del procés de desenvolupament, s'utilitza *Azure DevOps*, que ens proporciona un conjunt d'eines integrades per donar suport a la integració i entrega contínues (CI/CD) i la gestió àgil de projectes. La integració de *git* amb *Azure DevOps* ens permet utilitzar branques de desenvolupament, facilitar revisions de codi i fusions de branques, i automatitzar els processos de *building* i *deployment* per garantir desplegaments ràpids i segurs.

5 Eines utilitzades

Aquesta secció s'ha afegit per explicar millor les diferents eines utilitzades per al desenvolupament del planificador.
S'han seleccionat eines eficients i compatibles per a cada etapa del projecte. Destaquem les següents:

- **Next.js i React.js:** Base del desenvolupament del front-end, amb Next.js gestionant el *rendering* del costat del servidor i les API calls, i React.js per a interfícies dinàmiques i reactives, assegurant escalabilitat.
- **TypeScript:** Adoptat pel seu tipatge estàtic que millora la detecció d'errors en compilació, millorant la robustesa del codi.
- **Tailwind CSS:** Utilitzat per a un disseny *responsive* i personalitzat mitjançant un sistema de classes utilitàries.
- **Syncfusion:** Utilitza React Scheduler per a planificació, oferint components amb l'estil de Microsoft Fluent UI.
- **Azure i Azure DevOps:** Azure proveeix la infraestructura *cloud* i *hosting*, juntament amb Azure DevOps, que gestiona el repositori de codi.
- **Prisma:** ORM que facilita la modelització d'entitats i operacions CRUD amb una interfície intuïtiva.
- **Jira:** Eina de gestió àgil que utilitza un tauler Kanban per a l'organització i el seguiment de tasques.
- **Git:** Sistema de control de versions essencial per a la col·laboració segura i la gestió de canvis al codi font.

6 Planificació

S'ha eliminat la secció d'estratègies de desplegament. S'ha ajustat la descripció d'algunes fases de desenvolupament a la secció 6.1. El temps estimat de duració de diverses tasques a la secció 6.2 s'ha modificat, ja que s'han detectat desviacions segons la planificació inicial:
***1- Integració:** 3 → 5 setmanes*
***2- UI:** 5 → 6 setmanes*
***3- Forecast:** 5 → 4 setmanes*
***4- Proves:** 1 → 2 setmanes*
També s'ha modificat el diagrama de Gantt de l'Annex 1.
La planificació del projecte s'estructura al voltant d'un enfocament iteratiu i àgil, adaptant-se a les necessitats canviants dels diferents departaments. El procés de desenvolupament s'organitza en cicles de treball definits, amb cada cicle enfocat en el lliurament d'un conjunt específic de funcionalitats.

6.1 Fases de Desenvolupament

1. **Fase d'inici:** S'estableixen els fonaments del projecte, incloent-hi el recull de requisits, el disseny de la interfície d'usuari, la configuració de l'entorn de desenvolupament, la creació de repositoris a Azure DevOps, juntament amb la planificació inicial del *backlog* a Jira seguint el tauler Kanban.
2. **Fase de Desenvolupament Iteratiu:** En aquesta fase, s'implementen cicles de desenvolupament iteratius, on cada cicle inclou les etapes de Assigned, In Progress, Pending Info, Technical Validation, Functional Validation i Done. Aquesta fase es caracteritza pel desenvolupament continu de característiques, millores i correccions d'errors, amb revisions regulars per adaptar les prioritats segons calgui.
3. **Fase d'integració i proves:** Les funcionalitats desenvolupades són integrades i sotmeses a proves per assegurar-ne la qualitat i el rendiment. Aquesta fase també inclou proves d'acceptació de l'usuari per validar que el sistema compleix els requisits establerts. En aquesta fase, es fan les proves en un entorn de QA, dins d'una màquina virtual i apuntant a una URL privada.
4. **Fase de Lliurament i Desplegament:** Un cop validat el software, es procedeix al desplegament a l'entorn de producció. Aquest procés és recolzat per Azure DevOps juntament amb Azure App Service, facilitant la integració i entrega contínues (CI/CD) per automatitzar el desplegament i minimitzar els riscos associats.

6.2 Planificació de tasques

1.- Integració de Dades
Desenvolupar i implementar el sistema d'integració de dades entre plataformes com CIMSA i la base de dades del CMB amb l'aplicació.
- *Temps estimat:* 5 setmanes.
- *Criticitat:* Alta
- *Importància:* Alta

2.- Disseny i implementació de la UI
Crear una interfície d'usuari intuitiva i àgil, basada en l'ecosistema de Microsoft. Implementar les diferents vistes del planificador: dia, setmana, mes, any, agenda. Dissenyar i desenvolupar també les interfícies de planificació de tasques i projectes, edició de tasques, configuració del planificador, *login* i *signup*.
- *Temps estimat:* 6 setmanes.
- *Criticitat:* Crítica
- *Importància:* Alta

3.- Funcionalitats de *Forecasting*
Dissenyar i desenvolupar les eines de *forecasting* de producció estimada i real en format de taula editable. Es podrà editar la quantitat de producció d'un projecte (en %) i l'import corresponent per a assegurar una producció exacte.
- *Temps estimat:* 4 setmanes.
- *Criticitat:* Alta
- *Importància:* Mitja

4.- Proves de validació
Realitzar proves de validació i aprovació del *software* per assegurar la qualitat i correctesa del sistema, i l'acceptació de l'usuari i els *stakeholders*.
- *Temps estimat:* 2 setmanes.
- *Criticitat:* Mitja
- *Importància:* Mitja

5.- Desplegament i entrega
Desplegar l'aplicació a l'entorn de producció i realitzar l'entrega final. Comprovar que l'aplicació és accessible i recollir *feedback* dels usuaris per implementar millores i corregir errors.
- *Temps estimat:* 2 setmanes.
- *Criticitat:* Alta
- *Importància:* Crítica

Es pot trobar el diagrama de Gantt a l'*Apèndix A1*.

6.3 Seguiment del projecte

El seguiment i l'avaluació del progrés del projecte es fan a través d'informes de seguiment funcionals, lliurats semanal o mensualment. Això permet a l'equip i als *stakeholders* tenir una visió clara de l'estat del projecte previ a les reunions de seguiment.

Les reunions regulars de revisió i *feedback* s'utilitzen per avaluar el treball realitzat, identificar àrees de millora, i ajustar la planificació i estratègies segons calgui.

7 Anàlisis de riscos

Aquesta secció no s'ha modificat al primer informe de progrés.

Retards en la integració de dades	**Probabilitat** Mitja	**Impacte** Alt
Efecte: Pot causar retards en el desenvolupament posterior del projecte.	**Solució:** Realitzar anàlisis preliminars detallats i establir els models de les dades.	

Restriccions en el disseny i desenvolupament de la UI	**Probabilitat** Baixa	**Impacte** Mig
Efecte: Impacte en la usabilitat de la interfície i en l'acceptació de l'usuari final.	**Solució:** Establir prototips, organitzar l'arquitectura de components.	

Dificultats amb l'utilització d'APIs externes (corporatives)	**Probabilitat** Mitja	**Impacte** Alt
Efecte: Pot causar retards en el desenvolupament posterior del projecte.	**Solució:** Realitzar proves i definir l'estructura de les dades per a cada operació CRUD.	

Canvis als requeriments	**Probabilitat** Mitja	**Impacte** Alt
Efecte: Pot causar dificultats a la implementació del sistema, augmentar (o reduir, en casos excepcionals) el temps d'entrega.	**Solució:** Organitzar reunions setmanals o mensuals amb els diferents *stakeholders* per re-validar els requisits i documentar els canvis.	

Problemes al desplegament	**Probabilitat** Baixa	**Impacte** Crític
Efecte: Ineficiència dels laboratoris, descontentament dels usuaris finals i dels *stakeholders*.	**Solució:** Realitzar anàlisis preliminars detallats i establir els models de les dades	

8 Valoració de costos

S'han ajustat les durades de servei de la infraestructura software i les llicències (de N/A a 1 any).

Recurs	Temps	Cost	Cost Total
Project Manager Encarregat de connectar els diferents *stakeholders*, alinear els seus interessos i comunicar efectivament les decisions amb l'equip.	16 setmanes	35€ / hora	**22,400.00€**
Full Stack Developer Intern Encarregat de dissenyar, implementar i testejar l'aplicació.	16 setmanes	10€ / hora	**6,400€**
Técnic d'infraestructures Será qui dirigeixi el desplegament del *software* a l'entorn corporatiu.	2 setmanes	25€ / hora	**2,000€**
Infraestructura hardware Inclou una aproximació de costos relacionats amb dispositius, perifèrics, costos d'electricitat, etc.	16 setmanes	3€ / dia	**336€**
Infraestructura software Inclou les bases de dades i l'allotjament al núvol de l'aplicació.	*1 any*	25€ / mes	**300€**
Llicències de llibreries de components Inclou la llicència de Syncfusion.	*1 any*	1.000€ / any	**1,000€**
COST TOTAL APROXIMAT (1 ANY)			**32,436.00€**

9 Desenvolupament

Secció afegida al primer informe de seguiment.
El desenvolupament del projecte s'estructura al voltant de tres àrees principals: el *front-end*, el *back-end*, la base de dades i les proves. Aquest enfocament ens permet concentrar-nos en les especificacions tècniques i funcionals de cada component, assegurant un desenvolupament cohesiu i eficient del sistema.

9.1 Front-end

El desenvolupament del *front-end* es centra en crear una interfície d'usuari clara i intuïtiva, que pugui ser fàcilment utilitzada pels empleats d'Applus sense una corba d'aprenentatge pronunciada. Per això, s'han seleccionat Next.js i Tailwind CSS, a causa de la seva flexibilitat, eficiència en la renderització i el seu ampli suport. Tailwind CSS proporciona un marc de treball per al disseny ràpid i coherent de la UI, i facilita la implementació d'un disseny *responsive* que permet als usuaris planificar de forma còmode.

Components

Els components funcionals de React són una manera d'escriure components (visuals) com a funcions a React. Aquests components prenen les propietats com a argument i retornen elements React, que descriuen el que hauria d'aparèixer a la pantalla. Són diferents dels components de classe, que són més tradicionals a React i impliquen funcions més complexes com els mètodes de cicle de vida i la gestió de l'estat.

En un component funcional, simplement es crea una funció que retorna JSX (una extensió de sintaxi per a JavaScript que s'assembla a HTML), o en el cas del CMB, retorna TSX, que és el mateix però utilitzant el tipat estàtic de TypeScript, addicionalment. Per exemple:

```tsx
import React from 'react';

type PrimaryButtonProps = {
  label: string;
  onClick: () => void;
  type?: 'button' | 'submit' | 'reset';
  disabled?: boolean;
}

const PrimaryButton: React.FC<PrimaryButtonProps> = ({
  label,
  onClick,
  type = 'button',
  disabled = false
}) => {
  return (
    <button
      type={type}
      disabled={disabled}
      onClick={onClick}
      className="primary-button"
    >
      {label}
    </button>
  );
};

export default PrimaryButton;
```

Aquesta funció és un component React que accepta un argument de tipus *PrimaryButtonProps* i retorna un botó personalitzat i dinàmic, de tipus *React.FC*. Gràcies al tipat de les *props* del component i el seu *return*, evitem haver de fer proves de consistència de dades. Aquests components són funcions de primera classe a JavaScript i poden utilitzar altres funcions com ara *hooks* per gestionar l'estat i els efectes secundaris.

Alguns dels avantatges que ens proporcionen els components funcionals de React són:

1. **Simplicitat:** Els components funcionals són més fàcils de llegir i provar perquè només són funcions de JavaScript sense utilitzar la paraula clau "*this*".
2. ***Hooks*:** introduïts a React 16.8, els *hooks* permeten que els components funcionals gestionin l'estat, els efectes secundaris, el context i molt més, cosa que els fa tant, o més potents que els components de classe.
3. **Rendiment:** els components funcionals poden ser menys complicats i tenir un rendiment lleugerament millor que els components de classe, tot i que React ha optimitzat el rendiment per a tots dos tipus en actualitzacions recents.

D'altra banda, alguns dels desavantatges que presenten són:

1. **Corba d'aprenentatge per a *hooks*:** Entendre els *hooks* (com *useState*, *useEffect*) pot ser complex per als principiants.
2. ***Overhead* amb *re-renders* freqüents:** Si no es gestiona correctament amb *hooks* com *useMemo* i *useCallback*, els components funcionals poden provocar problemes de rendiment a causa de les repeticions freqüents.
3. **Gestió del cicle de vida menys intuïtiu:** La gestió dels mètodes del cicle de vida pot ser menys intuïtiu perquè requereix una bona comprensió dels *hooks*.

Complementàriament, Next.js és un marc de React que ofereix funcions com ara la renderització per la part del servidor i la gestió de rutes per directoris, el qual ha servit d'ajuda per a millorar el rendiment de l'aplicació. Es basa i amplia React, inclosos els components funcionals.

En el context del CMB, els components funcionals representen una part crítica de l'aplicació. Cada component es pot descriure com una part reutilitzable de la interfície visual. Pot ser un botó, una taula, o un component que serveixi de contenidor per a altres components. A la següent figura es pot veure la interfície principal del planificador desestructurada en components.

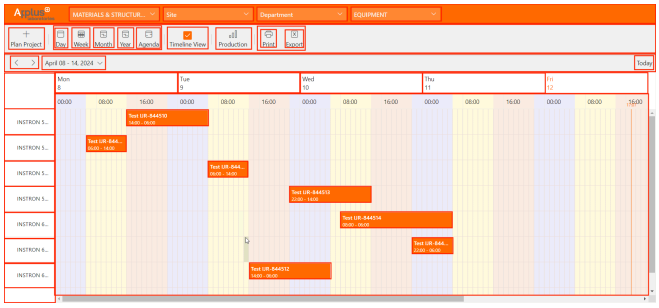


Fig. 3. Components funcionals a la vista principal

Lògica de processament

En el context dels projectes React i Next.js, tant les *utils* (funcions o mòduls d'utilitat) com els *custom hooks* són eines essencials per crear bases de codi més modulars, reutilitzables i organitzades. Aquests conceptes ajuden a gestionar la complexitat, especialment a mesura que creix l'escala i la funcionalitat de l'aplicació.

Les *utils* són funcions d'ajuda o mòduls que realitzen tasques habituals que no estan directament vinculades a la lògica d'interfície d'usuari de cap component específic. Aquestes funcions són generalment pures, és a dir, no modifiquen estats externs ni produeixen efectes secundaris. En canvi, prenen entrades, les processen i retornen sortides.

Alguns dels mòduls de *utils* del planificador serveixen per:

- Emmagatzemar totes les **constants** del codi. És a dir, valors que sempre seran iguals, i que no poden ser modificats.
- **Formatejar** les dades de les tasques a planificar rebudes desde les APIs corporatives a un format adequat al planificador i a la nostre base de dades.
- Realitzar el **setup** de l'aplicació: demanar totes les dades necessàries per a poder iniciar el planificador i poblar-lo amb les dades rebudes.

Els *custom hooks*, d'altra banda, són funcions que permeten emmagatzemar la lògica de processament d'un o més components funcionals, sempre que utilitzin les funcions d'estat i cicle de vida de React. Són una característica potent introduïda a React 16.8 que permet extreure la lògica dels components en funcions reutilitzables.

Un *custom hook* pot utilitzar altres *hooks* (com ara *useState*, *useEffect*, *useContext*, etc.) i proporcionar una manera de compartir lògica amb diferents components. El nom dels *custom hooks* normalment comença amb '*use*', per indicar que es tracta d'un *hook*, seguint la convenció i les regles dels *hooks* de React. Per exemple:

```
import { useState, useEffect } from 'react';

function useDebounce<T>(value: T, delay: number = 200):
T {
  const [debouncedValue, setDebouncedValue] =
useState<T>(value);

  useEffect(() => {
    const handler = setTimeout(() => {
      setDebouncedValue(value);
    }, delay);

    return () => {
      clearTimeout(handler);
    };
  }, [value, delay]);

  return debouncedValue;
}

export default useDebounce;
```

Aquest *custom hook* s'utilitza en diversos components del planificador per gestionar valors que necessitin un retard en la seva actualització, com entrades a formularis de búsqueda dinàmics o per evitar l'*spam* de clics a *checkboxes*.

Estilatge

L'ús de Tailwind CSS juntament amb React Functional Components aporta diversos avantatges a l'hora de dissenyar i estilar cada component:

1. **Enfocament *Utility-First***
   Tailwind CSS opera amb el principi *utility-first*, el que significa que ofereix classes d'utilitat de baix nivell que ajuden a crear dissenys complexos directament al components TSX. Aquest enfocament fa que sigui increïblement flexible i ràpid dissenyar la interfície d'usuari de l'aplicació.

2. **Configuració d'estils**
   El fitxer de configuració de Tailwind (*tailwind.config.js*) permet personalitzar el sistema de disseny (com ara colors, tipus de lletra, punts d'interrupció, etc.). Aquesta configuració de disseny centralitzada ajuda a mantenir la coherència en tot el projecte, semblant al sistema de context de React que comparteix estats entre components.

3. ***Responsiveness***
   Tailwind CSS inclou utilitats *responsive* molt àgils. Els components de React es poden fer *responsive* fàcilment mitjançant les utilitats de punt d'interrupció de Tailwind. Això fa que sigui senzill dissenyar components que s'adaptin a diferents mides de dispositius, millorant la capacitat de resposta de l'aplicació sense esforç addicional.

4. **Codi net i llegible**
   Com que Tailwind utilitza classes d'utilitat, el codi dels components funcionals de React es pot mantenir net i centrat. D'aquesta manera es podran entendre fàcilment quins estils s'apliquen directament al marcatge dels components, millorant la llegibilitat i facilitant el procés de depuració.

En el següent exemple, es pot veure l'utilització de Tailwind al component 'NavbarItem' del planificador:

```
<li
       className={`cursor-pointer transition-all rounded-md p-2
${liClassName}`}
     onClick={handleClick}
   >
     <Link
         className="flex gap-[10px] my-0 mx-[10px] items-center
relative"
       href={props.href ? props.href : ""}
       title={props.text}
     >
       {props.item}
       <p>{props.text}</p>
       {props.isLink ? (
                 <i  className="absolute  -right-[18px]  top-1/2
-translate-y-1/2 size-5 opacity-60">
           <LinkIcon />
         </i>
       ) : props.NavbarFunc.name === "expandNavbar" ? (
                 <i  className="absolute  -right-[18px]  top-1/2
-translate-y-1/2 size-5 opacity-60 overflow-hidden">
           <ExpandRightIcon />
         </i>
       ) : (
         ""
       )}
       {props.isLink ? (
                 <i  className="absolute  -right-[18px]  top-1/2
-translate-y-1/2 size-5 opacity-60">
           <LinkIcon />
         </i>
       ) : (
         ""
       )}
     </Link>
   </li>
```

9.2 Back-end

Al *back-end*, el focus està a garantir la robustesa, seguretat i escalabilitat del sistema. S'utilitza TypeScript per aprofitar els avantatges del tipat estàtic, cosa que resulta en un codi més segur i fàcil de mantenir. Les funcionalitats de gestió de consultes a la base de dades i la creació dels models de dades són facilitats per Prisma ORM i el sistema de rutes de Next.js 14 – *App Router*, que assegura una integració fluida i eficient entre el *back-end* i el *front-end*.

Una part crítica del *back-end* és el desenvolupament d'un sistema d'integració de dades capaç de sincronitzar i consolidar informació de diverses fonts internes d'Applus, utilitzant APIs dissenyades específicament per a aquest propòsit. Això no només inclou la sincronització de dades en temps real sinó també la implementació d'un sistema d'autorització i autenticació per garantir la seguretat i la privadesa de les dades.

Microserveis

El primer pas per adoptar una arquitectura de microserveis en aquest context és descompondre el sistema de consultes a la base de dades en serveis més petits i gestionables, cadascun responsable d'una part del negoci. Algunes de les funcions del planificador són:

- **Gestió de recursos:** s'encarrega de gestionar les operacions CRUD dels recursos, a més de gestionar consultes amb filtres.
- **Gestió de projectes:** organitza les consultes CRUD dels projectes i l'obtenció de les seves tasques.
- **Gestió de tasques:** permet les operacions CRUD de les tasques. Es pot veure a continuació:

```
export const getTasks = async () => {
 try {
   return await prismaCMB.tasks.findMany();
 } catch (error) {
     console.error("Could not get planned tasks collection: ",
error);
   throw error;
 }
};
export const deleteTask = async (id: string) => {
 try {
```

```
    return await prismaCMB.tasks.delete({ where: { id } });
  } catch (error) {
    console.error(`Could not delete task with id ${id}: `, error);
    throw error;
  }
};
export const createTask = async (data: Tasks) => {
  try {
    return await prismaCMB.tasks.create({ data });
  } catch (error) {
    console.error("Could not create task correctly: ", error);
    throw error;
  }
};
export const updateTask = async (id: string, data: Tasks) => {
  try {
    return await prismaCMB.tasks.update({ where: { id }, data });
  } catch (error) {
      console.error(`Could not update task ${id} correctly: `,
error);
    throw error;
  }
};
```

- **Gestió d'usuaris:** gestiona el *log-in* i la sessió de l'usuari.

Cada microservei té el propi context de domini i pot interactuar amb la base de dades de manera independent utilitzant el client Prisma. Aquest client de Prisma es configura a un únic fitxer, que conté un patró Singleton. D'aquesta manera, ens assegurem que només una instància de client de Prisma s'està executant a l'aplicació:

```
import { PrismaClient as CMBPrismaClient } from
"../../../database/cmb";
import { PrismaClient as DWHPrismaClient } from
"../../../database/dwh";

const CMBPrismaClientSingleton = () => {
  return new CMBPrismaClient();
};

const DWHPrismaClientSingleton = () => {
  return new DWHPrismaClient();
};

declare global {
  var prismaCMB: undefined | ReturnType<typeof
CMBPrismaClientSingleton>;
  var prismaDWH: undefined | ReturnType<typeof
DWHPrismaClientSingleton>;
}

export const prismaCMB = globalThis.prismaCMB ??
CMBPrismaClientSingleton();
export const prismaDWH = globalThis.prismaDWH ??
DWHPrismaClientSingleton();
```

En el codi anterior es pot apreciar un patró Singleton per a cadascun dels clients de Prisma que utilitza l'aplicació: el client PrismaCMB s'utilitza per a les consultes a la base de dades propia de l'aplicació, i el client PrismaDWH s'utilitza per a les consultes a la base de dades d'Applus QPS, la divisió d'Applus que pertany a Canadà i Estats Units.

Gràcies a aquesta aproximació, s'obtenen una sèrie de beneficis, descrits a continuació:

- **Escalabilitat:** Els microserveis poden escalar independentment en funció de la necessitat del recurs al que serveixen.
- **Resiliència:** La fallada en un microservei no afectarà als altres, millorant l'estabilitat general del sistema.
- **Manteniment:** Es facilita molt el procés de prova i manteniment, ja que cada servei ve encapsulat per defecte.

App Router

Amb la recent actualització de Next.js 14 i la nova característica *App Router* que implementa, s'ha pogut redefinir i optimitzar la manera com s'organitzen les pàgines i les APIs en aplicacions web, com en el cas del CMB. L'*App Router* ofereix una forma més flexible i potent de gestionar l'enrutament a Next.js, facilitant una estructura més neta i modular. Tot seguit, s'explica com s'ha aprofitat aquesta funcionalitat en el context del planificador.

1. Estructura d'Enrutament Simplificada
App Router permet definir rutes de manera programàtica, el que significa que es poden crear rutes més dinàmiques i configurables sense la necessitat d'estructurar els fitxers i carpetes sota el directori *'pages'*. Això és especialment útil en aplicacions grans i complexes com ara un planificador de laboratoris, on les rutes poden variar significativament i necessiten una configuració més flexible.

2. Gestió Integrada d'APIs
Un dels avantatges més significatius de l'*App Route*r és la seva capacitat per gestionar tant les pàgines com les rutes d'API des d'un sol lloc. Això ha permès organitzar millor el codi relacionat amb la interfície d'usuari i les interaccions de back-end. Per exemple, les APIs que gestionen les operacions CRUD de cada model de dades poden ser fàcilment definides i mantingudes juntament amb les rutes de les pàgines que les consumeixen, millorant la cohesió i la mantenibilitat del codi.

3. Middleware a Nivell de Ruta
L'*App Router* introdueix la possibilitat d'aplicar middleware específic a rutes individuals o grups de rutes. Al planificador, això s'utilitzarà per gestionar l'autenticació i l'autorització de manera més granular. Per exemple, certes rutes que accedeixen a pàgines privades poden tenir capes addicionals de seguretat que verifiquin els permisos de l'usuari abans de procedir amb l'operació sol·licitada.

4. Millora a l'Experiència de Desenvolupament
Finalment, l'ús de l'*App Router* ha millorat l'experiència de desenvolupament proporcionant un enfocament més intuïtiu i potent per a l'enrutament. S'han pogut aprofitar característiques com l'encaminament de tipus *safe*, que ajuda a detectar els errors relacionats amb rutes incorrectes en temps de compilació. Així aconseguim

reduir els errors i millorar la qualitat general del codi.

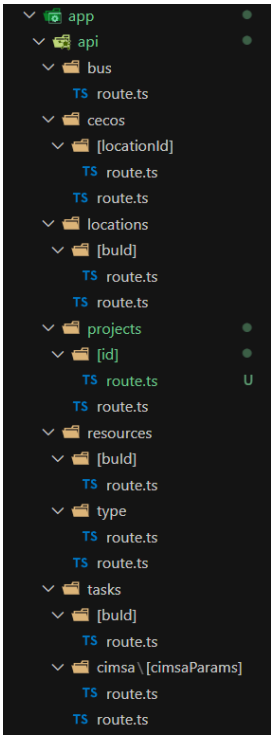


Fig.4. Rutes API amb App Router

9.3 Base de Dades

Per gestionar les dades eficientment, s'ha implementat una arquitectura de dades robusta utilitzant una combinació de Microsoft Azure SQL Server i els dos datalakes que ens proveiran les APIs de negoci: CIMSA i DWH (*Data WareHouse*). Aquesta arquitectura facilita la integració de dades a gran escala procedents de diverses fonts dins de l'organització. A continuació, descric com s'estructura aquesta implementació per maximitzar l'eficàcia i l'eficiència del sistema.

Microsoft Azure SQL Server

Microsoft Azure SQL Server és una base de dades relacional que proporciona un rendiment, una seguretat i una escalabilitat robustes per a aplicacions empresarials.

En el context del CMB, s'ha establert el model de dades de l'aplicació seguint el diagrama adjuntat a l'Annex 8, on es poden veure totes les taules i les diverses relacions entre elles.

API corporativa de CIMSA

CIMSA, el datalake central d'Applus, actua com un repositori centralitzat per a les tasques a planificar. La integració amb aquest datalake es fa mitjançant dues APIs corporatives.

La primera API s'encarrega de retornar les tasques a planificar d'un projecte en concret, a partir dels paràmetres:

- **Offer code:** Codi d'oferta del projecte (facturat al client).
- **Offer CRM code:** Codi d'oferta del projecte al CRM d'Applus.
- **Project ID:** ID del projecte a CIMSA.
- **Project SAP Code:** Codi del projecte al SAP d'Applus.

Si qualsevol dels paràmetres coincideix amb un projecte, l'API ens retornarà una llista d'objectes, aquest objectes representen cada tasca amb diverses tuples de tipus *'key: value'* en format JSON. Es pot veure el format més en detall a l'Annex 7.

La segona API és la responsable d'enviar el valor de la producció de cada tasca de volta a CIMSA, un cop s'ha generat la producció al planificador. Accepta dos paràmetres:

- **Task ID:** ID de la tasca a CIMSA.
- **Production amount:** Valor total, en euros o en dòlars, del que s'ha produït sobre la tasca, per tal de poder facturar-ho al client.

9.4 Proves

Les proves juguen un paper crucial en el desenvolupament del projecte, assegurant la qualitat i la fiabilitat del programari.

Afortunadament, a Applus contem amb un departament dedicat, a més de serveis subcontractats, que es dediquen a realitzar diversitat de proves funcionals, d'integració, etc. a les aplicacions desenvolupades pels diferents equips d'IT dins d'Applus Laboratories.

Gràcies a aquest servei, en el nostre cas ens podem centrar únicament en les proves de validació i verificació, col·laborant tant amb usuaris com amb els *stakeholders*.

10 Conclusió

S'ha modificat la conclusió completament. S'ha explicat millor l'experiència de desenvolupar un projecte com aquest i els aprenentatges obtinguts.

Com es va esmentar a l'inici d'aquest document, aquest treball no ha estat simplement un Treball de Fi de Grau; ha constituït un procés complet de desenvolupament de *software*, analitzat, dissenyat i implementat d'acord amb els estàndards d'un entorn empresarial real.

L'automatització de processos mitjançant les trucades a APIs ha estat el motor darrere de l'execució de tasques al llarg de cada etapa del cicle de vida del programari.

La fase de planificació va ser crucial i es va abordar a les etapes inicials, actualitzant-se paral·lelament a mesura que evolucionava el projecte. L'estimació inicial del temps necessari no va ser totalment precisa a causa de la meva manca d'experiència i de circumstàncies extraordinàries durant el desenvolupament. No obstant això, es van realitzar els canvis oportuns per assolir un resultat funcional i eficient:

- En certes fases, el temps estimat per a algunes tasques, especialment a la codificació dels processos d'integració de dades, va ser excessivament optimista.

- Les etapes del cicle de vida del programari no es van

seguir de manera estrictament cronològica com es detalla en aquest document. Va ser necessari tornar a la fase de disseny diverses vegades per resoldre ambigüitats i errors. En particular, la fase de definició de requisits i disseny es va modificar substancialment durant el desenvolupament del front-end per adaptar-se a les necessitats reals del projecte i els *stakeholders*.

En un pla personal, aquest projecte ha marcat el meu primer contacte amb el desplegament d'una solució integral de *software* empresarial especialitzat, la gestió i l'automatització de la integració contínua, i el monitoratge de la seva execució. Aquest procés m'ha permès aplicar els coneixements adquirits durant la meva especialització en Enginyeria del *Software*, com també tots els coneixements que he obtingut per compte propi durant els últims anys del grau.

Al llarg d'aquests mesos, el meu interès per les tecnologies *full-stack* ha crescut, descobrint un ampli rang de plataformes, *frameworks* i llibreries que faciliten la creació, desenvolupament, desplegament i monitorització de projectes. Aquest aspecte ha obert un ventall de possibilitats per a projectes futurs.

Entre els aspectes menys favorables, la planificació inicialment optimista va resultar ser un obstacle que va requerir una reorganització de tasques i una abstracció més gran del treball. La càrrega de treball inicial va ser subestimada fins a fases avançades del desenvolupament.

En avaluació personal, considero que l'exercici en aquest projecte ha estat satisfactori, i la motivació sostinguda durant aquest període ha estat decisiva. A nivell d'aprenentatge, aquest projecte ha representat un avanç significatiu en el meu desenvolupament professional i personal, consolidant la meva passió per la tecnologia i la innovació en l'enginyeria de *software*.

Agraïments

Aquesta secció no s'ha modificat al primer informe de progrés.
Voldria expressar el meu agraïment a totes les persones que han contribuït al desenvolupament i finalització d'aquest projecte. En primer lloc, al meu equip de treball, la dedicació, enginy i col·laboració del qual han estat fonamentals per superar els desafiaments i assolir els objectius plantejats.

També vull agrair-me a mi mateix, pel compromís, la perseverança i l'esforç invertit en aquest projecte. Ha estat un camí ple d'aprenentatge i creixement personal i professional, on cada obstacle superat ha reforçat la meva passió pel *software* i la innovació tecnològica.

La meva gratitud s'estén també a Marc Talló, el meu tutor, per la seva orientació, suport i consells valuosos al llarg d'aquest procés. La seva experiència i coneixement han estat de gran ajuda.

Finalment, agraeixo a Applus+ i a tots els seus membres per brindar-me l'oportunitat de realitzar les meves pràctiques a la seva divisió de Laboratoris. Aquesta experiència ha estat crucial per al desenvolupament del meu projecte i ha proporcionat un context real i valuós per aplicar els coneixements adquirits durant la meva formació al grau d'Enginyeria Informàtica a la UAB.

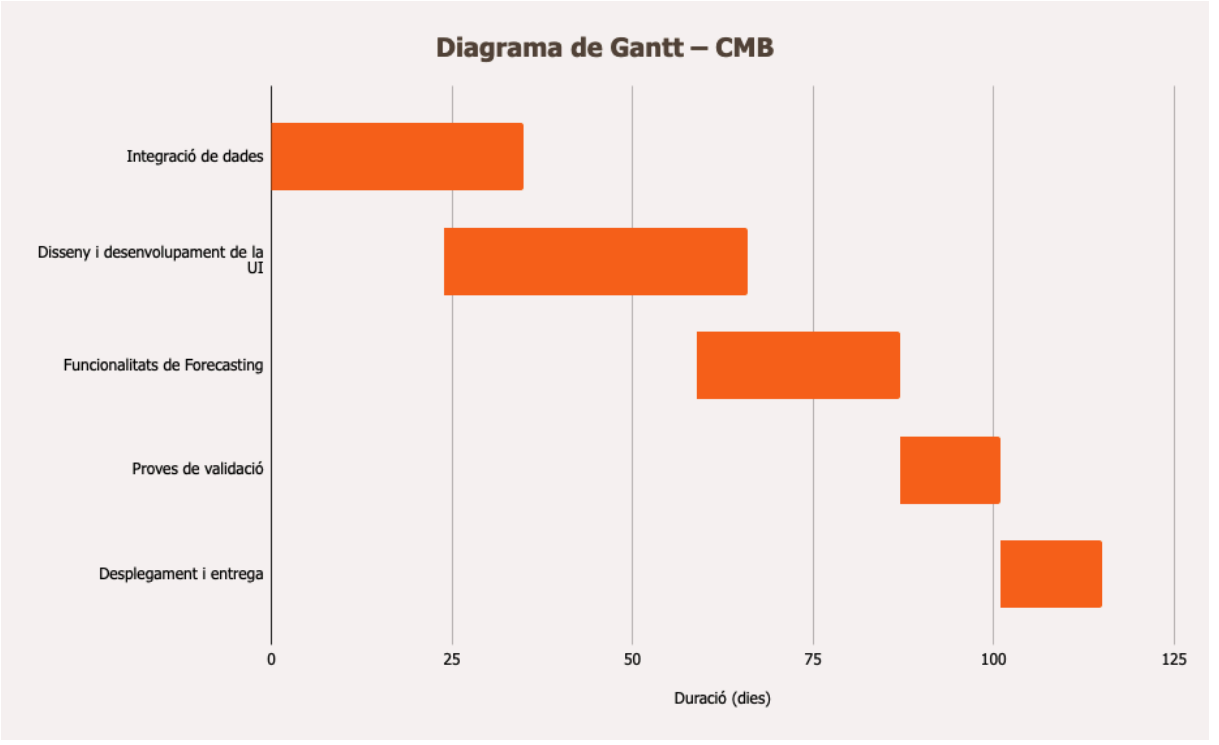
Bibliografia

S'han explicat breument els links per a una millor comprensió i s'han afegit alguns de nous.

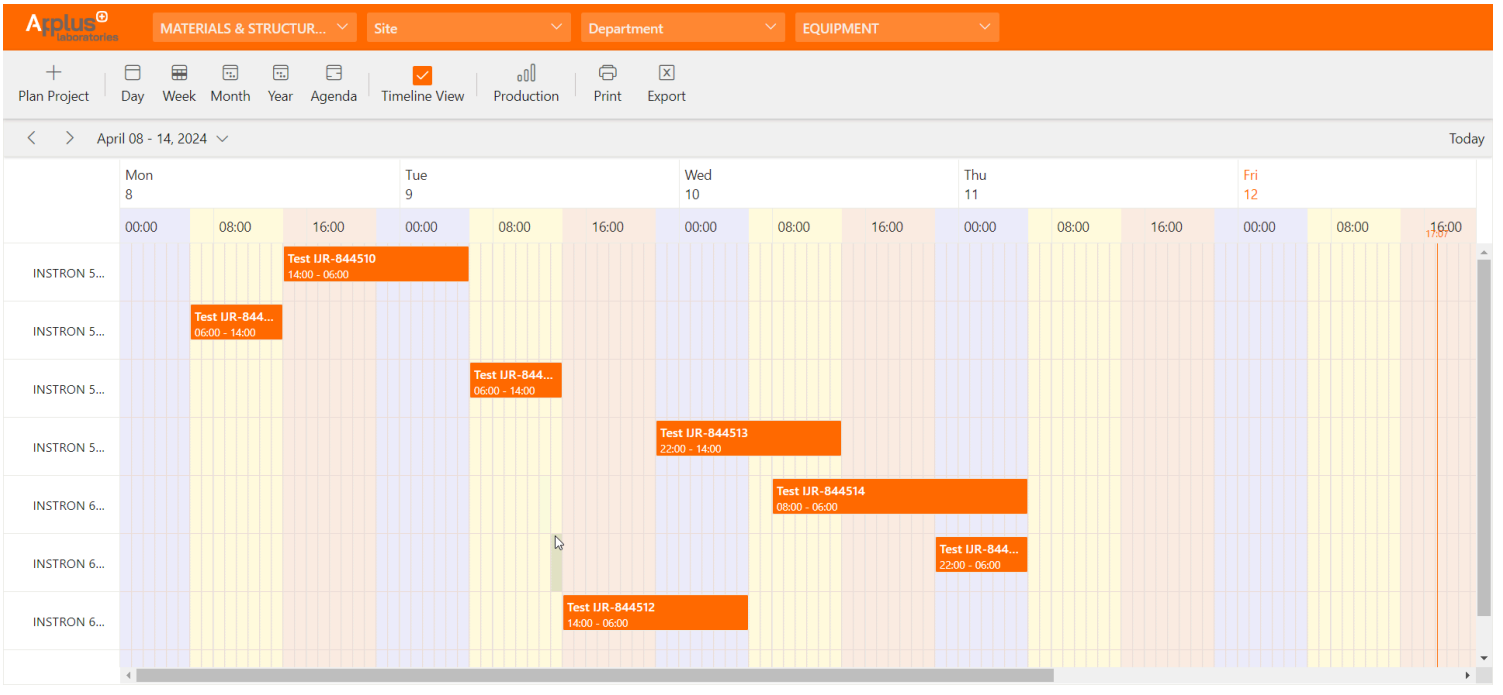
[1] Applus+ Laboratories | Informació sobre Applus+: https://www.appluslaboratories.com/global/en/
[2] Testing, Inspection and Certification – TIC Council | Què és el sector TIC?: https://www.tic-council.org/about-us/what-is-the-tic-sector
[3] Next.js | Javascript Full-stack Framework: https://nextjs.org/
[4] React.js | Javascript Front-end Framework: https://react.dev/
[5] TypeScript | Llibreria de tipat estàtic per Javascript: https://www.typescriptlang.org/
[6] Tailwind CSS | Llibreria de components UI i classes personalitzades de CSS: https://tailwindcss.com/
[7] Microsoft Azure | Cloud de Micrososft: https://azure.microsoft.com/en-us
[8] Prisma ORM | Llibreria de models relacionals de mapejat d'objectes: https://www.prisma.io/
[9] CRUD APIs – AppMaster | Què són les APIs CRUD?: https://appmaster.io/es/glossary/api-crud-crear-leer-actualizar-eliminar
[10] Jira – Atlassian | Software de planificiació: https://www.atlassian.com/software/jira
[11] Kanban – Atlassian | Eina de planificació de Jira: https://www.atlassian.com/agile/kanban
[12] Git | Eina de control de versions: https://git-scm.com/
[13] Azure DevOps | Plataforma per gestionar repositoris de Microsoft: https://azure.microsoft.com/en-us/products/devops
[14] Azure App Service | Servei d'aplicacions web de Microsoft Azure: https://azure.microsoft.com/es-es/products/app-service
[15] CI/CD – Red Hat | Què és el CI/CD?: https://www.redhat.com/en/topics/devops/what-is-ci-cd
[16] Syncfusion React | Llibreria de components React per a planificació: https://www.syncfusion.com/react-components
[17] Next.js 14 App Router | Documentació oficial sobre l'App Router: https://nextjs.org/docs/app
[18] React *Custom Hooks* | Com es pot reutilitzar lògica amb els *custom hooks*? https://react.dev/learn/reusing-logic-with-custom-hooks

APÈNDIX

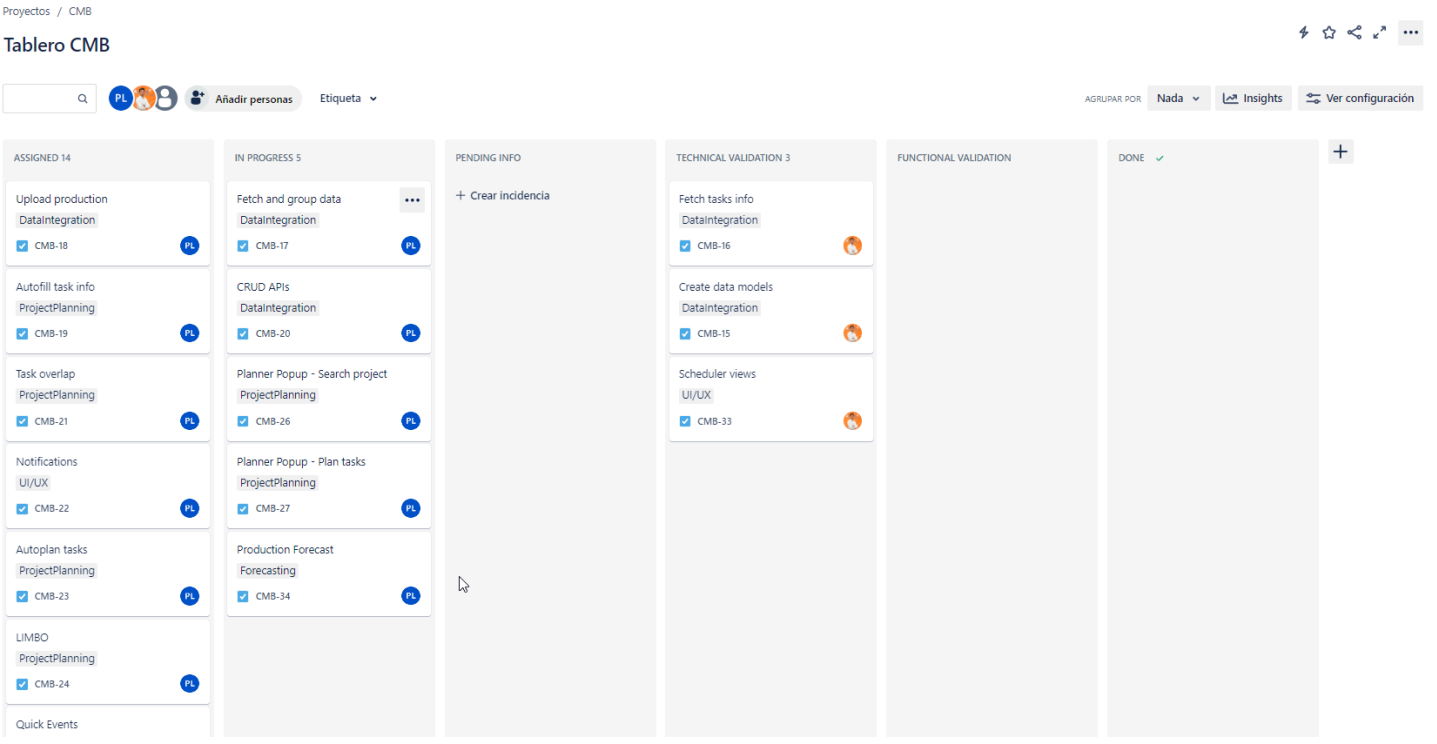
A1. Diagrama de Gantt



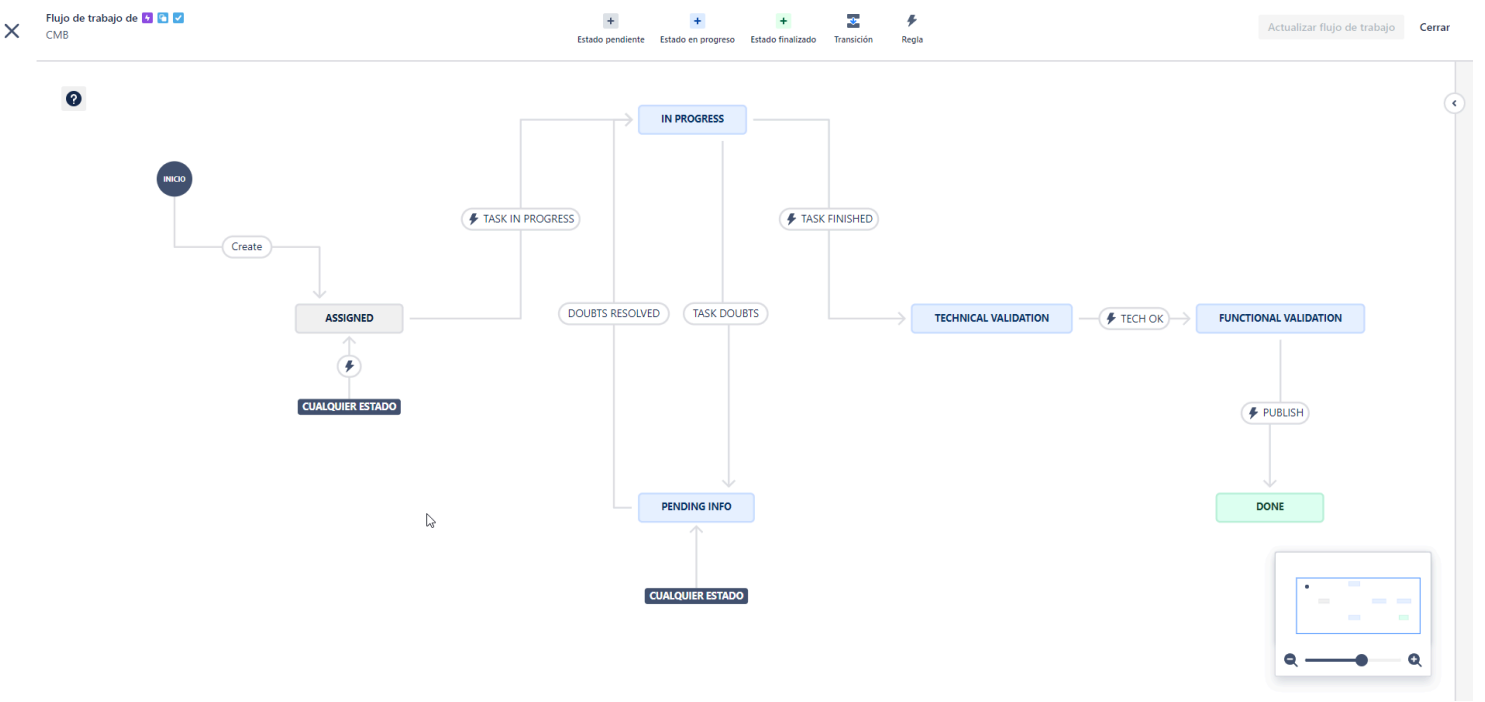
A2. Vista principal del CMB



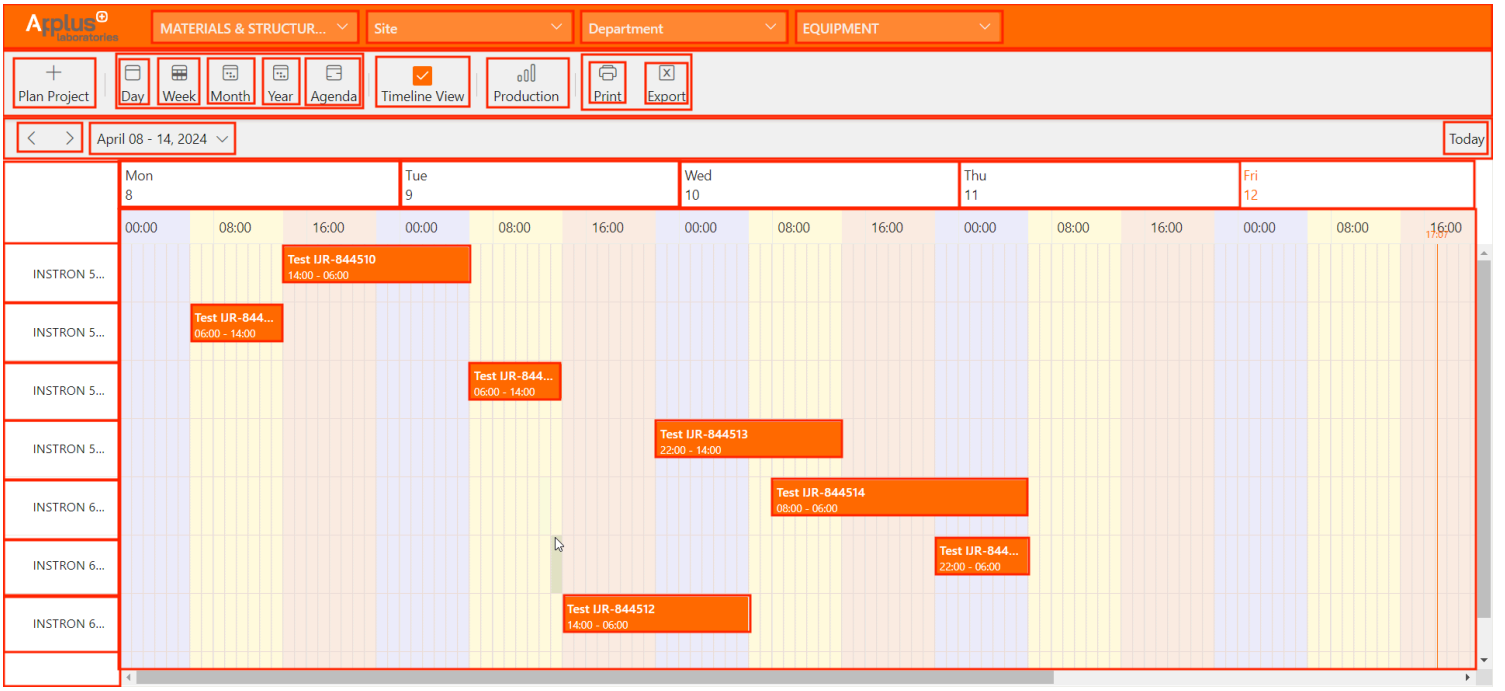
A3. Taulell Kanban



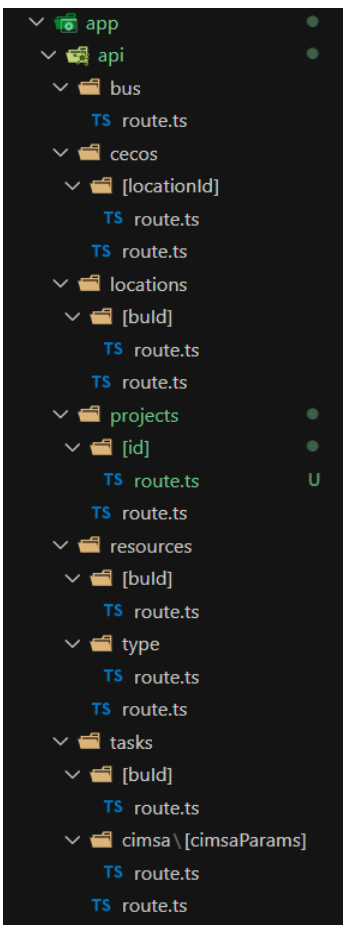
A4. Configuració del taulell Kanban



A5. Components funcionals a la vista principal



A6. Rutes API del planificador amb l'*App Router* de Next.js 14



A7. Objecte retornat per l'API de CIMSA

```ts
type CimsaTask = {
 offerCode: string;
 codeWithVersion?: string;
 offerCrmCode: string;
 projectId: number;
 projectSapCode: string;
 taskGroupId: number;
 taskGroupName: string;
 taskGroupDescription: string;
 taskGroupAmount: number;
 taskGroupTypeName: string;
 taskGroupCostCenterCode: string;
 taskGroupCostCenterName: string;
 taskGroupProductCode: string;
 taskGroupProductName: string;
 taskGroupTotalClosedProductionAmount: number;
 taskGroupTotalOpenProductionAmount: number;
 taskGroupTotalBillingAmount: number;
 taskId: number;
 taskPersonHours: number;
 taskEquipmentHours: number;
};
```

A8. Diagrama de la BBDD