Implementació d'una arquitectura escalable en una aplicació web enfocat al negoci de la pastisseria

Frank Calderón Molina

Resum — En aquest article s'exposa el desenvolupament realitzat amb l'objectiu de construir una arquitectura de microserveis escalable enfocada al negoci de la pastisseria. Aquest desenvolupament té com a objectiu optimitzar els processos que es duen a terme en les pastisseries, permetre gestionar tant negocis petits com grans empreses, aportar flexibilitat al negoci per poder-se adaptar a un mercat cada cop més digital i que canvia ràpidament, i facilitar l'obtenció de dades per poder prendre decisions estratègiques de manera més eficient. Es tracta d'un projecte desenvolupat al voltant de quinze setmanes de treball utilitzant una metodologia waterfall o cascada, seguint les fases d'anàlisi, disseny, implementació, desplegament i proves, i tancament. Al final del projecte, l'objectiu és tenir implementada una arquitectura escalable en Kubernetes que ofereixi a l'usuari la capacitat de gestionar productes i ingredients, el seu estoc i les comandes que reben a la pastisseria.

Paraules clau — Pastisseria, gestió, estoc, productes, ingredients, comandes, Kubernetes, Docker, arquitectura, cloud, escalabilitat, microservei

Abstract — This article presents the development carried out with the objective of building a scalable microservices architecture focused on the bakery business. The goal of this development is to optimize the processes that take place in bakeries, allowing management of both small and large businesses, providing flexibility to adapt to an increasingly digital and rapidly changing market, and facilitating efficient data acquisition for strategic decision-making. The project was developed over approximately fifteen weeks using a waterfall methodology, following the phases of analysis, design, implementation, deployment and testing, and closure. At the end of the project, the objective is to have implemented a scalable architecture in Kubernetes that offers users the ability to manage products and ingredients, their stock, and the orders received by the bakery.

Index Terms — Bakery, management, stock, products, ingredients, orders, Kubernetes, Docker, architecture, cloud, scalability, microservice

1 Introducció - Context del trebal

El sector de la pastisseria, en general, és un sector tradicional format per negocis familiars que han aconseguit sobreviure al llarg dels anys, però que han quedat estancats en l'àmbit tecnològic. En analitzar els processos diaris que duen a terme, es poden trobar maneres d'optimitzar-los a través del software. La gestió de l'estoc, l'organització de les comandes i la gestió de les finances són exemples clars d'aquests processos que es poden optimitzar mitjançant eines digitals.

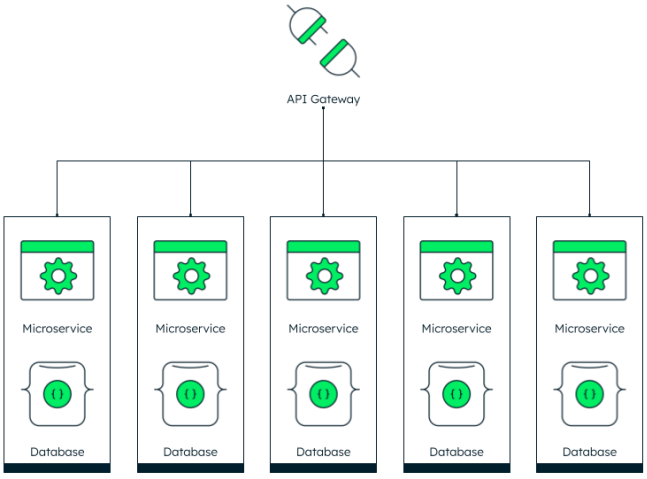
Aquest projecte té com a objectiu resoldre aquest problema d'optimització en el sector de la pastisseria, amb la finalitat d'incrementar la productivitat. Això permetria que pastisseries més petites puguin fer front a empreses més grans i, en conseqüència, augmentar els ingressos del negoci.

2 Objectius

L'objectiu principal del projecte és desenvolupar un software en format API [1] centrat en la gestió de les comandes, els productes, els ingredients i l'estoc d'un negoci de pastisseria. A part, també ha de ser escalable, és a dir, escalar els seus recursos en funció de la demanda del software. Per fer-ho es desenvoluparà una arquitectura de microserveis.

Tal com es pot observar en la imatge 1, una arquitectura de microserveis es basa en dividir el software en múltiples peces de software, coneguts com a serveis o microserveis, les quals són independents unes de les altres.

- *E-mail de contacte: frankcalderonmolina@gmail.com*
- *Menció realitzada: Enginyeria del Softwars*
- *Treball tutoritzat per: Victoria Montenegro Ruíz (Àrea de Ciències de la Computació i Intel·ligència Artificial)*
- *Curs 2022/23*



Imatge 1: Arquitectura de microserveis

Com a base d'aquesta arquitectura tenim una API gateway [2] o porta d'enllaç i un conjunt de microserveis.
Una API gateway [2] o porta d'enllaç és una peça de software encarregada de comunicar l'usuari que vol utilitzar el software amb els diferents microserveis que el conformen i de retornar les respostes dels microserveis a l'usuari.
Un microservei [3] és una peça de software encarregada de realitzar una funció concreta. Per exemple, en una botiga online, podríem tenir diferents microserveis per realitzar funcions concretes com mostrar el catàleg de productes, administrar la cistella de compra, aplicar descomptes, etc.
Aquest tipus d'arquitectura ens permet desenvolupar software més flexible i que continuï funcionant malgrat que una part no ho pugui fer a causa d'un error.
A part, d'aquests dos elements bàsics de l'arquitectura, també podem tenir altres elements que ens ajudaran a realitzar funcions concretes. En el nostre cas l'arquitectura que desenvoluparem estarà formada pels següents elements:

- **Microservei per la gestió de les comandes:** Peça de software encarregada d'oferir a l'usuari les funcionalitats necessàries per gestionar les comandes.
- **Microservei per la gestió dels productes i els ingredients:** Peça de software encarregada d'oferir a l'usuari les funcionalitats necessàries per gestionar els productes i els ingredients.
- **Microservei per la gestió de l'estoc:** Peça de software encarregada d'oferir a l'usuari les funcionalitats necessàries per gestionar l'estoc dels ingredients.
- **Servidor de porta d'enllaç:** Peça de software encarregada de realitzar una comunicació bidireccional entre l'usuari i els diferents microserveis que conformen l'arquitectura.
- **Servidor d'autenticació:** Peça de software encarregada de gestionar la seguretat i l'autenticació dels usuaris per evitar que un usuari no registrat en el sistema pugui utilitzar les funcionalitats oferides per aquest.

Els objectius de desenvolupar una arquitectura d'aquest tipus per una pastisseria són:

- Optimitzar l'eficiència dels processos que es duen a terme en el negoci de la pastisseria mitjançant el desenvolupament d'un sistema informàtic.
- Potenciar el creixement del negoci a través d'un sistema escalable que permeti gestionar grans volums de dades.
- Aportar flexibilitat al negoci de manera que es puguin adaptar ràpidament als canvis en el mercat i a les necessitats del client.
- Facilitar el monitoratge i l'anàlisi de dades de l'empresa per obtenir informació sobre les necessitats dels clients.

En definitiva, l'objectiu és desenvolupar un software que implementi una arquitectura que permeti dur a terme certs processos que es duen a terme en les pastisseries, que permeti gestionar tant negocis petits com grans empreses, que aporti flexibilitat al negoci per poder-se adaptar a un mercat cada cop més digital i que canvia ràpidament, i que faciliti l'obtenció de dades per poder prendre decisions estratègiques de negoci.

3 Estat de l'art

En aquest apartat es destacaran eines similars al projecte desenvolupat, per tal de tenir un punt de referència respecte a les funcionalitats necessàries.
Cal destacar que, les eines presentades en aquest apartat, són utilitzades per empreses del sector en altres països. Realitzant una cerca, es fan servir principalment dues eines, Cybake (https://cybake.com) [4] i BakeSmart (https://bakesmart.com) [5].
Tot i així, no són eines utilitzades per empreses del nostre territori, però poden servir de referència de cara al desenvolupament del projecte.

3.1 Cybake

En paraules de l'empresa, Cybake [4] és un sistema de programari de gestió de fleques utilitzat per a fleques minoristes i al detall per controlar les seves operacions, reduir l'administració, millorar l'eficiència i augmentar les vendes.
Cybake [4] ofereix diverses funcionalitats als seus usuaris, entre les quals destaquen eines per gestionar la producció, funcionalitats relacionades amb punts de venda, integracions amb diverses plataformes, eines de business intelligence, etc.

3.2 BakeSmart

En paraules de l'empresa, BakeSmart [5] és un programari per a fleques creat específicament per satisfer les necessitats d'una fleca minorista.
BakeSmart [5] ofereix diverses funcionalitats als seus usuaris, entre les quals destaquen funcionalitats per vendre pastissos personalitzats, integració amb plataformes de comerç electrònic, eines per la gestió d'inventari i producció, etc.

4 Metodologia

La metodologia de treball que se seguirà, pel que fa al desenvolupament del projecte, és una metodologia de tipus cascada o waterfall [6]. S'ha seleccionat aquesta metodologia degut a la naturalesa del projecte, ja que s'ha de desenvolupar per una sola persona, amb el qual no es pot començar una fase sense haver acabat l'anterior. Aquesta

metodologia està composta per les següents fases:

- **Anàlisi:** Fase en la qual es realitza la planificació del projecte, es recullen els requisits dels usuaris i es defineixen les funcionalitats a desenvolupar durant el transcurs del projecte.
- **Disseny:** Fase en la qual es dissenya l'arquitectura del sistema i s'especifiquen els elements que el componen.
- **Implementació:** Fase en la qual es desenvolupen els diversos elements que componen el projecte i es comprova la seva qualitat mitjançant un conjunt de proves.
- **Desplegament i proves:** Fase en la qual es despleguen els elements desenvolupats i es comprova la qualitat de la integració entre ells mitjançant un conjunt de proves.
- **Tancament:** Fase en la qual es tanca el projecte.

Per tal d'organitzar la planificació i tenir una visió global de l'estat del projecte s'utilitzarà un tauler de l'eina Trello [7]. Aquest tauler estarà compost per les següents categories:

- **Backlog:** Tasques a realitzar durant el projecte.
- **To do:** Tasques a realitzar durant la fase del projecte.
- **In progress:** Tasques en les quals s'està treballant en aquell moment.
- **QA:** Tasques acabades de desenvolupar a l'espera de la seva integració amb les diferents parts acabades i de la seva revisió de qualitat.
- **Done:** Tasques finalitzades i integrades.

5 Planificació

Respecte a la planificació del projecte, s'han distribuït les fases al llarg de quinze setmanes, començant el projecte el 27 de febrer de 2023 i finalitzant-lo l'11 de juny de 2023.

La fase d'anàlisi recull la mateixa planificació del projecte i l'anàlisi dels requisits. Aquesta fase té una duració d'una setmana.

La fase de disseny recull el disseny de l'arquitectura que es desenvoluparà durant la fase d'implementació i la definició i especificació dels diversos casos d'ús que conformen el projecte. Aquesta fase té una duració de dues setmanes.

La fase d'implementació recull el desenvolupament i implementació dels elements transversals de l'arquitectura, així com els tres microserveis que la conformen. Aquesta fase té una duració de deu setmanes.

La fase de desplegament i proves recull el desplegament de l'arquitectura del projecte, les proves end-to-end del projecte i les proves de rendiment. Aquesta fase té una duració de dues setmanes.

La fase de tancament recull el tancament del projecte i una avaluació interna per determinar què es pot millorar de cara a futurs desenvolupaments. Aquesta fase té una duració d'una setmana i se superposa amb la darrera setmana de la fase de desplegament i proves.

6 Desenvolupament del projecte

En aquest apartat es detallen les diverses fases del projecte, així com els reptes que han suposat cadascuna d'elles i els resultats que obtinguts.

6.1 Fase d'anàlisi

Durant la fase d'anàlisi [6] del projecte s'ha portat a cap la planificació del projecte, la qual es pot observar en el diagrama de Gantt [8] a l'apèndix 2. A més, s'han recollit els requisits, tant funcionals [9] com no funcionals [10] del projecte, els quals es poden veure a l'apèndix 1.

6.2 Fase de disseny

Durant la fase de disseny [6] del projecte s'ha realitzat l'especificació dels diversos casos d'ús [11] segons els requisits recollits a la fase anterior.

A continuació es presenten els casos d'ús definits del projecte.

ID	Cas d'ús	Fet
CU01	Veure totes les comandes	Si
CU02	Veure les dades d'una comanda	Si
CU03	Crear una comanda	Si
CU04	Modificar una comanda	Si
CU05	Eliminar una comanda	Si
CU06	Preparar una comanda	Si
CU07	Finalitzar una comanda	Si
CU08	Entregar una comanda	Si
CU09	Cancel·lar una comanda	Si
CU10	Tornar una comanda a l'estat Per fer	Si
CU11	Veure tots els productes	Si
CU12	Veure les dades d'un producte	Si
CU13	Crear un producte	Si
CU14	Modificar un producte	Si
CU15	Eliminar un producte	Si
CU16	Afegir productes	Si
CU17	Consumir productes	Si
CU18	Veure tots els ingredients	Si
CU19	Veure les dades d'un ingredient	Si
CU20	Crear un ingredient	Si
CU21	Modificar un ingredient	Si
CU22	Eliminar un ingredient	Si
CU23	Afegir estoc d'un ingredient	Si
CU24	Consumir estoc d'un ingredient	Si
CU25	Veure tots els estocs	Si
CU26	Veure les dades d'un estoc	Si
CU27	Crear un estoc	Si
CU28	Modificar un estoc	Si
CU29	Eliminar un estoc	Si
CU30	Caducar un estoc	Si

Taula 1: Casos d'ús del projecte

A més a més, s'ha dissenyat el diagrama que representa l'estructura de l'arquitectura del projecte.

Aquest diagrama té com a objectiu representar de manera visual l'estructura dels components que componen l'arquitectura i les interaccions que hi ha entre ells.

Aquest diagrama es pot trobar a l'apèndix 3.

A més a més, s'ha dissenyat un altre diagrama que representa el flux de les peticions HTTP des de que entren a l'arquitectura fins que es dona una resposta a l'usuari.

El diagrama del flux de les peticions es pot trobar a l'apèndix 4.

Pel que fa al flux de les peticions una vegada l'arquitectura ja ha iniciat, es conforma de set passos:

1. El client realitza una petició contra el servidor d'autenticació amb les seves credencials.
2. El servidor d'autenticació retorna al client un codi (token) d'accés.
3. El client fa una petició contra l'API gateway [2] incloent el codi (token) d'accés obtingut en iniciar sessió.
4. L'API gateway [2] redirigeix la petició del client cap al microservei [3].
5. El microservei valida el codi (token) d'accés del client contra el servidor d'autenticació.
6. El microservei [3] retorna la resposta a l'API gateway [2].
7. L'API gateway [2] retorna la resposta al client.

A part, també s'ha dissenyat un diagrama d'estats [12] que representa el flux principal del projecte, és a dir, el flux de l'estat de les comandes.

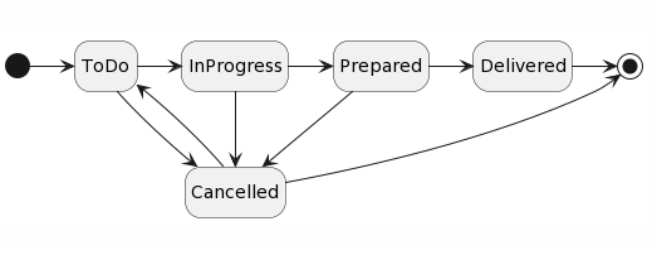
Tenint en compte que el projecte s'acaba basant pràcticament en gestionar comandes s'ha definit un diagrama d'estats [12] que representa les transicions d'estat que pot tenir una comanda.

En primer lloc, una comanda comença en l'estat "Per fer" (To Do) en el moment de la seva creació. Una vegada aquí, la comanda pot passar a l'estat "En progrés" (In Progress) en el moment en què es comenci a preparar, o a l'estat "Cancel·lada" (Cancelled) si ha estat cancel·lada.

En segon lloc, una vegada es troba en l'estat "En progrés" (In Progress), la comanda pot passar a l'estat "Preparada" (Prepared) un cop s'hagi finalitzat la seva elaboració, o a l'estat "Cancel·lada" (Cancelled) si ha estat cancel·lada.

En tercer lloc, una vegada es troba en l'estat "Preparada" (Prepared), la comanda pot passar a l'estat "Entregada" (Delivered) un cop s'hagi entregat al client, o a l'estat "Cancel·lada" (Cancelled) si ha estat cancel·lada.

Finalment, una vegada es troba en l'estat "Cancel·lada" (Cancelled), la comanda pot tornar a l'estat inicial "Per fer" (To Do).



Imatge 2: Diagrama d'estats de les comandes

6.3 Fase d'implementació

Durant la fase d'implementació [6] del projecte, s'han desenvolupant els elements que conformen l'arquitectura, així com els elements necessaris per poder desplegar-los dins de l'arquitectura en la fase de desplegament i proves.

Per una banda, s'han desenvolupat els elements que conformen l'arquitectura, els quals són: servidor d'autenticació, API gateway, servidor de descobriment, microservei per la gestió dels productes i els ingredients, microservei per la gestió de l'estoc i microservei per la gestió de les comandes.

Per altra banda, s'han desenvolupat les definicions dels diferents elements de Kubernetes [13] necessaris per poder posar en marxa l'arquitectura.

En primer lloc, s'ha implementat el servidor d'autenticació, pel qual s'ha seleccionat l'eina Keycloak [14]. Keycloak és una peça de software que ens permetrà realitzar la funció de servidor d'autenticació. Com a servidor d'autenticació, la seva funció principal serà la de permetre als usuaris enregistrar-se per, un cop iniciada la sessió, poder accedir a les funcionalitats dels microserveis [3].

Ja que es tracta d'una eina ja desenvolupada, l'únic que s'ha requerit ha estat integrar-lo en l'arquitectura de Kubernetes [13] que s'està desenvolupant mitjançant un recurs de tipus Deploy [15] (encarregat de desplegar l'eina dins de l'arquitectura), un recurs de tipus Service [16] (encarregat de fer que l'eina sigui accessible) i un recurs de tipus Ingress [18] (encarregat de fer que l'eina sigui accessible des de fora de l'arquitectura a través de l'URL http://auth.sweetify.com).

En segon lloc, s'ha desenvolupat l'API gateway, el qual es basa en un projecte Java [19] utilitzant el framework Spring [20]. Gràcies a aquest framework [21], podem definir un API gateway [2] que redirigeixi als microserveis [3] que hi hagin registrats al servidor de descobriment.

Per poder fer aquest desenvolupament s'han necessitat les següents dependències:

- **spring-cloud-starter-gateway:** Transforma el projecte en un API gateway [3] que pot obtenir dades del servidor de descobriment.
- **spring-cloud-starter-kubernetes-fabric8-all:** Permet que el projecte obtingui els paràmetres de configuració d'un recurs de Kubernetes de tipus ConfigMap.

En aquest cas, el desenvolupament només comporta afegir les dependències especificades anteriorment al projecte i configurar-lo a través de les propietats de configuració.

Una vegada desenvolupat el projecte, s'ha integrat en l'arquitectura de Kubernetes [13] que s'està desenvolupant mitjançant els següents recursos:

- Un recurs de tipus Deploy [15]: Encarregat de desplegar l'eina dins de l'arquitectura.
- Un recurs de tipus Service [16]: Encarregat de fer que l'eina sigui accessible.
- Un recurs de tipus ConfigMap [17]: Encarregat d'emmagatzemar les propietats de configuració.
- Un recurs de tipus Ingress [18]: Encarregat de fer que l'eina sigui accessible des de fora de l'arquitectura a través de l'URL http://api.sweetify.com.

En tercer lloc, s'ha desenvolupat el servidor de descobriment, el qual es basa en un projecte Java [19] utilitzant el framework Spring [20]. Gràcies a aquest framework [21], podem definir un servidor de descobriment que s'encarregarà d'enregistrar els serveis que apuntin cap a ell.

Per poder fer aquest desenvolupament s'han necessitat les següents dependències:

- spring-cloud-starter-netflix-eureka-server: Transforma el projecte en un servidor de descobriment.

Igual que en el cas anterior, el desenvolupament només comporta afegir les dependències especificades anteriorment al projecte i configurar-lo a través de les propietats

de configuració.
Una vegada desenvolupat el projecte, s'ha integrat en l'arquitectura de Kubernetes [13] que s'està desenvolupant mitjançant els següents recursos:

- Un recurs de tipus Deploy [15]: Encarregat de desplegar l'eina dins de l'arquitectura.
- Un recurs de tipus Service [16]: Encarregat de fer que l'eina sigui accessible.
- Un recurs de tipus Ingress [18]: Encarregat de fer que l'eina sigui accessible des de fora de l'arquitectura a través de l'URL http://discovery.sweetify.com.

Cal destacar que, finalment, aquest element ha quedat descartat de l'arquitectura per evitar que el servidor de porta d'enllaç depengui de l'estat d'aquest component per realitzar la seva funció d'encaminament. A part, el fet de tenir aquest component desplegat afegia un pas innecessari en el desplegament de l'arquitectura, ja que els diversos components es connectaven a ell per enregistrar-se.
Aquest component ha estat substituït per una configuració dins del desenvolupament del servidor de porta d'enllaç o API gateway [2].
En quart lloc, s'ha desenvolupat el servidor de configuració, el qual es basa en un projecte Java [19] amb el framework Spring [20] en la seva versió 6. Gràcies a aquest framework [21], podem definir un servidor de configuració que permeti als diversos serveis que apuntin cap a ell configurar-se. Les propietats que necessiten els serveis per configurar-se estan emmagatzemades en un repositori, el qual és accedit pel servidor de configuració.
Per poder fer aquest desenvolupament s'han necessitat les següents dependències:

- spring-cloud-config-server: Transforma el projecte en un servidor de configuració.

Una vegada desenvolupat el projecte, s'ha integrat en l'arquitectura de Kubernetes [13] que s'està desenvolupant mitjançant els següents recursos:

- Un recurs de tipus Deploy [15]: Encarregat de desplegar l'eina dins de l'arquitectura.
- Un recurs de tipus Service [16]: Encarregat de fer que l'eina sigui accessible.
- Un recurs de tipus Ingress [18]: Encarregat de fer que l'eina sigui accessible des de fora de l'arquitectura a través de l'URL http://config.sweetify.com.

Cal destacar que, finalment, aquest element ha quedat descartat de l'arquitectura per evitar que els desenvolupaments depenguin de l'estat d'aquest component per desplegar-se. A part, el fet d'haver-se de connectar al servidor de configuració abans de desplegar-se, afegia un pas innecessari en el desplegament de l'arquitectura.
En el seu lloc s'ha decidit utilitzar recursos de Kubernetes de tipus ConfigMap [17] per emmagatzemar la configuració dels diversos elements que conformen l'arquitectura.
En cinquè lloc, s'ha desenvolupat el microservei per la gestió dels productes i els ingredients, ja que no té cap dependència amb els altres microserveis. Aquest projecte es basa en un projecte Java [19] utilitzant el framework Spring [20].
Per dur a terme aquest microservei [3] s'han desenvolupat els següents elements:

- Models: Classes Java [19] encarregades de transformar les definicions de les taules de la base de dades en objectes de programació. S'han desenvolupat tres models, un per cadascuna de les taules de la base de dades.
- Repositoris: Classes Java [19] encarregades d'interaccionar amb la base de dades. S'han desenvolupat tres repositoris, un per cadascuna de les taules de la base de dades.
- Serveis: Classes Java [19] encarregades de dur a terme els casos d'ús definits en punts anteriors invocant funcions dels repositoris. S'han desenvolupat dos serveis, un per dur a terme els casos d'ús relacionats amb els productes i un per dur a terme els casos d'ús relacionats amb els ingredients.
- Controladors: Classes Java [19] encarregades d'exposar rutes HTTP per oferir les funcionalitats desenvolupades als serveis. S'han desenvolupat dos controladors, un per exposar les funcionalitats del servei de productes i un per exposar les funcionalitats del servei d'ingredients.
- DTOs: Classes Java [19] encarregades de transformar les dades rebudes pel microservei en objectes de programació, ja sigui a través de les peticions HTTP que realitza l'usuari o a través de les consultes a la base de dades. S'han desenvolupat set DTOs, un per cada grup de dades diferent que s'han hagut de transformar.
- Excepcions: Classes Java [19] encarregades de gestionar les excepcions produïdes pel microservei, per exemple, en els casos en què no es trobi el producte o ingredient al qual s'està intentant accedir. S'han desenvolupat tres excepcions diferents, a part de la lògica necessària per poder-les utilitzar.
- Configuracions: Classes Java [19] encarregades de definir certs paràmetres del comportament del microservei. En aquest cas s'han definit dues classes de configuració, ambdues relacionades amb la seguretat d'aquest. Una està destinada a implementar la seguretat del microservei, mentre que l'altre està destinada a anul·lar aquesta seguretat durant el desenvolupament.

A part, també s'han necessitat les següents dependències:

- spring-boot-starter-web: Permet definir un controlador de tipus REST al projecte amb diferents rutes a les quals accedir per obtenir els recursos.
- spring-boot-starter-data-jpa: Permet que el projecte pugui accedir a una base de dades i interpretar-la en base als models definits.
- spring-boot-starter-security: Permet que el projecte pugui implementar seguretat per accedir als recursos.
- spring-boot-starter-oauth2-resource-server: Permet que el projecte pugui implementar seguretat per accedir als recursos.
- spring-boot-starter-validation: Permet que el projecte pugui implementar validacions de dades.
- postgresql: Permet que el projecte pugui interactuar amb una base de dades de tipus PostgreSQL [22].
- spring-cloud-starter-kubernetes-fabric8-all: Permet que el projecte obtingui les propietats de configuració definides dins d'una arquitectura Kubernetes [13].
- jackson-databind: Permet realitzar serialització de

dades.

A més a més, degut a les necessitats del projecte, s'ha dissenyat i implementat una base de dades per poder emmagatzemar la informació necessària pel microservei [3].

Imatge 3: Base de dades de productes i ingredients

Dins d'aquesta base de dades, podem veure una taula "products" on s'emmagatzema la informació relativa als productes, una taula "ingredients" on s'emmagatzema la informació relativa als ingredients, i una taula "products_ingredients" on s'emmagatzema la informació relativa a la relació entre productes i ingredients.

Per exemple, en el cas que vulguem definir un producte (producte_1) amb ID 1, que conté 2.5 unitats d'un ingredient (ingredient_1) amb ID 1, i 0.75 unitats d'un ingredient (ingredient_2) amb ID 2, podríem veure la següent informació a la taula "products_ingredients":

ID	Product_ID	Ingredient_ID	Quantity
1	1	1	2.5
2	1	2	0.75

Taula 2: Exemple de la base de dades de productes i ingredients

Pel que fa a la fiabilitat del microservei s'han definit un conjunt de tests unitaris per tal d'establir que el comportament d'aquest és l'esperat. S'han realitzat tests pels repositoris, serveis i controladors, en els quals, indirectament, també es comprova el comportament dels models, dels DTOs i de les excepcions.Addicionalment, també s'han definit un conjunt de proves utilitzant Postman [23], per tal de realitzar proves de caixa negra al microservei i poder comprovar el seu comportament un cop estigui integrat dins de l'arquitectura.

En sisè lloc, degut a la dependència amb el microservei per la gestió dels productes i els ingredients, s'ha desenvolupat el microservei per la gestió de l'estoc. Aquest projecte es basa en un projecte Java [19] utilitzant el framework Spring [20].

Per dur a terme aquest microservei [3] s'han desenvolupat els següents elements:

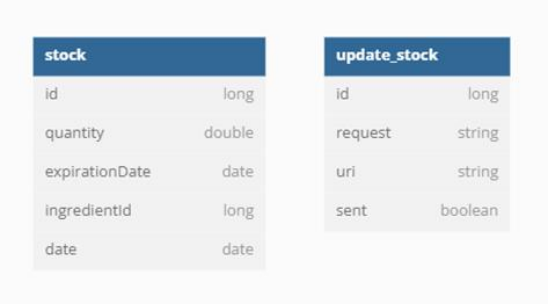
- Models: Classes Java [19] encarregades de transformar les definicions de les taules de la base de dades en objectes de programació. S'han desenvolupat tres models, dos corresponents a taules de la base de dades i un corresponent a un conjunt de dades utilitzat pel microservei.
- Repositoris: Classes Java [19] encarregades d'interaccionar amb la base de dades. S'han desenvolupat dos repositoris, un per cadascuna de les taules de la base de dades.
- Serveis: Classes Java [19] encarregades de dur a terme els casos d'ús definits en punts anteriors invocant funcions dels repositoris. S'han desenvolupat dos serveis, un per dur a terme els casos d'ús relacionats amb la gestió de l'estoc i un per dur a terme els casos d'ús relacionats amb l'interacció amb altres microserveis.
- Controladors: Classes Java [19] encarregades d'exposar rutes HTTP per oferir les funcionalitats desenvolupades als serveis. S'ha desenvolupat un controlador per exposar les funcionalitats del servei esmentat en el punt anterior.
- DTOs: Classes Java [19] encarregades de transformar les dades rebudes pel microservei en objectes de programació, ja sigui a través de les peticions HTTP que realitza l'usuari o a través de les consultes a la base de dades. S'han desenvolupat dos DTOs, un per cada grup de dades diferent que s'han hagut de transformar.
- Excepcions: Classes Java [19] encarregades de gestionar les excepcions produïdes pel microservei. En aquest cas només s'ha desenvolupat una excepció, en el cas que no es trobi l'estoc al qual s'està intentant accedir, a part de la lògica necessària per poder-la utilitzar.
- Configuracions: Classes Java [19] encarregades de definir certs paràmetres del comportament del microservei. En aquest cas s'han definit dues classes de configuració, ambdues relacionades amb la seguretat d'aquest. Una està destinada a implementar la seguretat del microservei, mentre que l'altre està destinada a anul·lar aquesta seguretat durant el desenvolupament.
- Utils: Eines d'ús comú per diverses classes del projecte. En aquest cas s'ha definit un DateTimeFormatter per gestionar les dates amb un mateix format.
- Scheduler: Classes Java [19] encarregades d'executar una funció concreta en un moment determinat cada cert temps. En aquest cas s'ha definit un component per executar dues funcions. Per una banda, podem trobar una funció encarregada de realitzar una petició d'actualització d'estoc al microservei per la gestió de productes i ingredients, i per l'altra banda podem trobar una funció encarregada d'expirar l'estoc caducat.

A part, també s'han necessitat les següents dependències:

- spring-boot-starter-web: Permet definir un controlador de tipus REST al projecte amb diferents rutes a les quals accedir per obtenir els recursos.
- spring-boot-starter-data-jpa: Permet que el projecte pugui accedir a una base de dades i interpretar-la en base als models definits.
- spring-boot-starter-security: Permet que el projecte pugui implementar seguretat per accedir als recursos.
- spring-boot-starter-oauth2-resource-server: Permet que el projecte pugui implementar seguretat per accedir als recursos.
- spring-boot-starter-validation: Permet que el projecte pugui implementar validacions de dades.
- postgresql: Permet que el projecte pugui interactuar amb una base de dades de tipus PostgreSQL [22].
- spring-cloud-starter-kubernetes-fabric8-all:

Permet que el projecte obtingui les propietats de configuració definides dins d'una arquitectura Kubernetes [13].
- jackson-databind: Permet realitzar serialització de dades.

A més a més, degut a les necessitats del projecte, s'ha dissenyat i implementat una base de dades per poder emmagatzemar la informació necessària pel microservei [3].



Imatge 4: Base de dades d'estoc

Pel que fa a la fiabilitat del microservei s'han definit un conjunt de tests unitaris per tal d'establir que el comportament d'aquest és l'esperat. S'han realitzat tests pels repositoris, serveis i controladors, en els quals, indirectament, també es comprova el comportament dels models, dels DTOs i de les excepcions.

Addicionalment, també s'han definit un conjunt de proves utilitzant Postman [23], per tal de realitzar proves de caixa negra al microservei i poder comprovar el seu comportament un cop estigui integrat dins de l'arquitectura.

Per acabar, degut a la dependència amb el microservei per la gestió dels productes i els ingredients, s'ha desenvolupat el microservei per la gestió de les comandes. Aquest projecte es basa en un projecte Java [19] utilitzant el framework Spring [20].

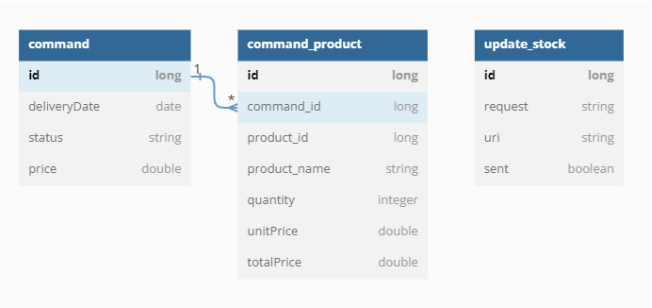
Per dur a terme aquest microservei [3] s'han desenvolupat els següents elements:

- Models: Classes Java [19] encarregades de transformar les definicions de les taules de la base de dades en objectes de programació. S'han desenvolupat quatre models, tres corresponents a taules de la base de dades i un corresponent a un conjunt de dades utilitzat pel microservei.
- Repositoris: Classes Java [19] encarregades d'interaccionar amb la base de dades. S'han desenvolupat tres repositoris, un per cadascuna de les taules de la base de dades.
- Serveis: Classes Java [19] encarregades de dur a terme els casos d'ús definits en punts anteriors invocant funcions dels repositoris. S'han desenvolupat dos serveis, un per dur a terme els casos d'ús relacionats amb la gestió de les comandes i un per dur a terme els casos d'ús relacionats amb l'interacció amb altres microserveis.
- Controladors: Classes Java [19] encarregades d'exposar rutes HTTP per oferir les funcionalitats desenvolupades als serveis. S'ha desenvolupat un controlador per exposar les funcionalitats del servei esmentat en el punt anterior.
- DTOs: Classes Java [19] encarregades de transformar les dades rebudes pel microservei en objectes de programació, ja sigui a través de les peticions HTTP que realitza l'usuari o a través de les consultes a la base de dades. S'han desenvolupat dos DTOs, un per cada grup de dades diferent que s'han hagut de transformar.
- Excepcions: Classes Java [19] encarregades de gestionar les excepcions produïdes pel microservei. En aquest cas s'han desenvolupat dues excepcions, una pel cas que no es trobi la comanda a la qual s'està intentant accedir, i una altra pel cas en que una comanda no pugui ser actualitzada. A part, s'ha desenvolupat la lògica necessària per poder-les utilitzar.
- Configuracions: Classes Java [19] encarregades de definir certs paràmetres del comportament del microservei. En aquest cas s'han definit dues classes de configuració, ambdues relacionades amb la seguretat d'aquest. Una està destinada a implementar la seguretat del microservei, mentre que l'altre està destinada a anul·lar aquesta seguretat durant el desenvolupament.
- Utils: Eines d'ús comú per diverses classes del projecte. En aquest cas s'ha definit un DateTimeFormatter per gestionar les dates amb un mateix format.
- Scheduler: Classes Java [19] encarregades d'executar una funció concreta en un moment determinat cada cert temps. En aquest cas s'ha definit un component per executar una funció funció encarregada de realitzar una petició d'actualització d'estoc al microservei per la gestió de productes i ingredients.

A part, també s'han necessitat les següents dependències:

- spring-boot-starter-web: Permet definir un controlador de tipus REST al projecte amb diferents rutes a les quals accedir per obtenir els recursos.
- spring-boot-starter-data-jpa: Permet que el projecte pugui accedir a una base de dades i interpretar-la en base als models definits.
- spring-boot-starter-security: Permet que el projecte pugui implementar seguretat per accedir als recursos.
- spring-boot-starter-oauth2-resource-server: Permet que el projecte pugui implementar seguretat per accedir als recursos.
- spring-boot-starter-validation: Permet que el projecte pugui implementar validacions de dades.
- postgresql: Permet que el projecte pugui interactuar amb una base de dades de tipus PostgreSQL [22].
- spring-cloud-starter-kubernetes-fabric8-all: Permet que el projecte obtingui les propietats de configuració definides dins d'una arquitectura Kubernetes [13].
- jackson-databind: Permet realitzar serialització de dades.

A més a més, degut a les necessitats del projecte, s'ha dissenyat i implementat una base de dades per poder emmagatzemar la informació necessària pel microservei [3].



Imatge 5: Base de dades de comandes

Pel que fa a la fiabilitat del microservei s'han definit un conjunt de tests unitaris per tal d'establir que el comportament d'aquest és l'esperat. S'han realitzat tests pels repositoris, serveis i controladors, en els quals, indirectament, també es comprova el comportament dels models, dels DTOs i de les excepcions.
Addicionalment, també s'han definit un conjunt de proves utilitzant Postman [23], per tal de realitzar proves de caixa negra al microservei i poder comprovar el seu comportament un cop estigui integrat dins de l'arquitectura.
Finalment, després de completar els desenvolupaments dels elements anteriors, s'han desenvolupat les definicions dels elements de Kubernetes [13] necessaris perquè l'arquitectura funcioni correctament.
Els elements de Kubernetes [13] definits són:

- Namespace: Fa referència a un clúster virtual.
- Deploy: Element encarregat de posar en marxa una imatge Docker [25] dins de l'arquitectura Kubernetes [13].
- Service: Element encarregat d'exposar un recurs de tipus Deploy dins de l'arquitectura Kubernetes [13].
- PersistentVolume: Element encarregat de reservar un espai de memòria.
- PersistentVolumeClaim: Element encarregat de vincular un recurs de tipus PersistentVolume amb un recurs de tipus Deploy.
- ConfigMap: Element encarregat de definir i emmagatzemar un conjunt de dades clau-valor.
- IngressController: Element encarregat d'exposar a l'exterior de l'arquitectura els recursos de tipus Service.
- ServiceAccount: Element encarregat de definir un compte d'usuari.
- ClusterRole: Element encarregat de definir un rol dins d'un clúster de Kubernetes.
- ClusterRoleBinding: Element encarregat de vincular un recurs de tipus ServiceAccount amb un recurs de tipus ClusterRole.
- HorizontalPodAutoscaler: Element encarregat de gestionar l'escalabilitat dels contenidors de Kubernetes.

Pel desenvolupament d'aquest projecte s'han definit els següents recursos:

- Un Namespace anomenat "Sweetify".
- Un Deploy per cadascun dels elements de l'arquitectura (elements transversals, microserveis i bases de dades).
- Un Service per cadascun dels elements de l'arquitectura (elements transversals, microserveis i bases de dades).
- Tres PersistentVolume, un per cadascuna de les bases de dades dels microserveis.
- Tres PersistentVolumeClaim, un per cadascuna de les bases de dades dels microserveis.
- Un ConfigMap pels següents elements: API gateway [3], servidor de descobriment, cadascun dels microserveis i cadascuna de les bases de dades.
- Un IngressController per definir les diferents rutes exposades a l'exterior de l'arquitectura.
- Un ServiceAccount, un ClusterRole i uun ClusterRoleBinding necessaris perquè els microserveis puguin accedir a les seves propietats de configuració definides als seus ConfigMaps [17].
- Un HorizontalPodAutoscaler [24] per cadascun dels microserveis, ja que són els elements de l'arquitectura que han d'escalar els seus recursos en funció de la demanda.

6.4 Fase de desplegament i proves

Durant la fase de desplegament i proves [6] del projecte, s'han executat les diverses definicions d'elements de Kubernetes [13] per posar en marxa l'arquitectura.
A més a més, s'han executat els diversos conjunts de tests de Postman [23] per comprovar les funcionalitats dels microserveis desenvolupats.
Per concloure aquesta fase, s'han executat diverses rondes de peticions contra els microserveis per comprovar que el seu rendiment sigui l'esperat.

6.5 Fase de tancament

Durant la fase de tancament [6] del projecte, s'ha dut a terme el procés de finalització del projecte i l'avaluació dels resultats obtinguts. Durant aquesta fase, s'ha realitzat una anàlisi exhaustiva de l'èxit de la implementació i s'han tancat totes les tasques i activitats relacionades amb el projecte.

7 RESULTATS

Com a resultats del projecte de desenvolupament d'una arquitectura de microserveis enfocat al negoci de la pastisseria cal destacar els següents punts: elements desenvolupats durant el transcurs del projecte i aportacions realitzades al projecte.

7.2 Elements desenvolupats

Durant el transcurs d'aquest projecte s'ha desenvolupat una arquitectura que conté els següents elements:

- **Microservei per la gestió de les comandes:** Peça de software encarregada d'oferir a l'usuari les funcionalitats necessàries per gestionar les comandes.
- **Microservei per la gestió dels productes i els ingredients:** Peça de software encarregada d'oferir a l'usuari les funcionalitats necessàries per gestionar els productes i els ingredients.
- **Microservei per la gestió de l'estoc:** Peça de software encarregada d'oferir a l'usuari les funcionalitats necessàries per gestionar l'estoc dels ingredients.
- **Servidor de porta d'enllaç:** Peça de software encarregada de realitzar una comunicació bidireccional entre

l'usuari i els diferents microserveis que conformen l'arquitectura.
- **Servidor d'autenticació:** Peça de software encarregada de gestionar la seguretat i l'autenticació dels usuaris per evitar que un usuari no registrat en el sistema pugui utilitzar les funcionalitats oferides per aquest.

7.2 Aportacions realitzades al projecte

Durant el transcurs d'aquest projecte s'han realitzat les següents aportacions al projecte:
- **Anàlisi i disseny de l'arquitectura de microserveis:** S'ha realitzat un anàlisi exhaustiu de les necessitats del negoci i s'ha dissenyat l'arquitectura de microserveis més adequada per cobrir aquestes necessitats.
- **Anàlisi i especificació de casos d'ús:** S'ha realitzat un anàlisi exhaustiu de les necessitats del negoci i s'han especificat els casos d'ús necessaris per cobrir aquestes necessitats.
- **Implementació de l'arquitectura:** S'ha implementat l'arquitectura dissenyada per cobrir les necessitats del negoci.
- **Proves:** S'han realitzat un conjunt de proves per avaluar la fiabilitat de l'arquitectura.

8 Conclusió

Com a conclusions del projecte, considero que ha estat una experiència molt enriquidora a nivell personal, ja que m'ha permitit guanyar coneixements i experiència tant en gestió de projectes, com a nivell tècnic.
Pel que fa a coneixements sobre gestió de projectes, considero que els més destacables, personalment, són els següents:
- **Planificació:** Al realitzar un projecte la planificació és un dels factors clau, i encara més si el projecte l'ha de realitzar una sola persona. Considero que tenir experiència en la planificació de projectes és molt important, ja no sols per projectes tecnològics, sinó en l'àmbit personal.
- **Gestió del temps i autoorganització:** Un altre dels factors clau al realitzar un projecte és la gestió del temps i, lligada a aquest, l'autoorganització. Considero que és crucial gestionar bé el temps disponible per intentar assolir els objectius proposats segons la planificació establerta per tal d'evitar retards o objectius no assolits.
- **Presa de decisions:** Durant el transcurs d'aquest projecte, un altre dels factors clau ha estat la presa de decisions. Al enfrontar-te sol a un projecte has de tenir el coratge de prendre decisions i assumir les conseqüències d'aquesta. Per exemple, quan no tens temps per perfeccionar un desenvolupament o per implementar totes les proves que voldries, has de prioritzar les tasques a realitzar per tal d'assolir l'objectiu del projecte.
- **Resolució de problemes:** Relacionat amb el punt anterior, un altre dels factors clau ha estat la resolució de problemes. Durant el desenvolupament d'un projecte, i més un projecte en solitari, s'ha de tenir una actitud proactiva, sobretot en la resolució de problemes, ja que no et pots recolzar en ningú per solucionar-lo.

Pel que fa a coneixements tècnics, considero que els més destacables, personalment, són els següents:
- Utilització del framework Spring [20] pel desenvolupament de microserveis i dels elements que conformen aquesta arquitectura.
- Coneixement sobre l'arquitectura de microserveis i els elements que la conformen.
- Utilització de Docker [25] per l'encapsulament de serveis.
- Utilització de Kubernetes per la gestió de contenidors Docker [25].
- Implementació i utilització d'un servei extern d'autenticació i autorització com Keycloak [14].

9 Treball futur

Durant el transcurs del projecte s'ha desenvolupat una arquitectura escalable que integra diversos microserveis per cobrir les necessitats de gestió de les comandes, productes, ingredients i estoc d'un negoci de pastisseria.
Aquesta arquitectura té diverses aplicacions i possibles extensions, incloent-hi:
- **Integració amb altres sistemes:** Es pot integrar amb altres sistemes, com ara sistemes de gestió de pagaments o plataformes de comerç electrònic, per a oferir la possibilitat de compra de productes.
- **Ampliació de funcionalitats:** Es poden afegir més funcionalitats als microserveis existents, com ara un sistema de valoracions i comentaris de productes o la integració amb xarxes socials per a una major visibilitat del negoci.
- **Integració amb una interfície frontal:** Es pot interactuar amb l'arquitectura desenvolupada a través d'una interfície frontal que sigui amigable per l'usuari per tal de que pugui fer servir les eines disponibles.

Agraïments

Voldria expressar el meu agraïment a la tutora del projecte, Victoria Montenegro, per la seva orientació i suport durant la realització del projecte. Gràcies a la seva guia el projecte s'ha pogut completar amb èxit.
També voldria donar les gràcies als meus amics i familiars per estar al meu costat, donar-me suport i entendre les hores dedicades a aquest projecte. El seu suport ha estat essencial per l'èxit del projecte.

Bibliografia

[1] ¿Qué es una API y cómo funciona? (s. f.). Red Hat. Visitat el 12/03/2023. URL: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces
[2] ¿Qué es API Gateway? (s. f.). TIBCO Software. Visitat el 12/03/2023. URL: https://www.tibco.com/es/reference-center/what-is-an-api-gateway
[3] ¿Qué son los microservicios? | AWS. (s. f.). Amazon Web Services, Inc. Visitat el 12/03/2023. URL: https://aws.amazon.com/es/microservices/
[4] Cybake® Bakery Software | Bakery Management Software.

(2023, 7 marzo). Cybake. Visitat el 02/03/2023. URL: https://cybake.com

[5] Bakery Software | BakeSmart. (2021). BakeSmart.com. Visitat el 02/03/2023. URL: https://bakesmart.com

[6] Metodología Waterfall: la gestión de proyectos en cascada - IEP. (2022, 7 julio). Instituto Europeo de Posgrado. Visitat el 12/03/2023. URL: https://www.iep.edu.es/metodologia-waterfall/

[7] Gestiona los proyectos de tu equipo desde cualquier lugar | Trello. (s. f.). Visitat el 12/03/2023. URL: https://trello.com/es

[8] (s. f.). Diagrama de Gantt: qué es y cómo crear uno con ejemplos. Asana. Visitat el 12/03/2023. URL: https://asana.com/es/resources/gantt-chart-basics

[9] Jain, A. (2023). Qué son los requisitos funcionales: ejemplos, definición, guía completa. Visure Solutions. Visitat el 03/03/2023. URL: https://visuresolutions.com/es/blog/functional-requirements/

[10] Jain, A. (2023). Qué son los requisitos no funcionales: ejemplos, definición, guía completa. Visure Solutions. Visitat el 03/03/2023. URL: https://visuresolutions.com/es/blog/non-functional-requirements/

[11] Definición de casos de uso. (2021, 3 setiembre). IBM. Visitat el 15/03/2023. URL: https://www.ibm.com/docs/es/elms/elm/6.0.3?topic=requirements-defining-use-cases

[12] Elements UML. (s. f.). Visitat el 05/04/2023. URL: https://docs.kde.org/trunk5/ca/umbrello/umbrello/uml-elements.html

[13] Orquestación de contenedores para producción. (s. f.). Kubernetes. Visitat el 12/03/2023. URL: https://kubernetes.io/es/

[14] Team, K. (s. f.). Keycloak. Visitat el 01/04/2023. URL: https://www.keycloak.org/

[15] Deployment. (2021, 16 agost). Kubernetes. Visitat el 01/04/2023. URL: https://kubernetes.io/es/docs/concepts/workloads/controllers/deployment/

[16] Service. (2022, 18 juliol). Kubernetes. Visitat el 01/04/2023. URL: https://kubernetes.io/es/docs/concepts/services-networking/service/

[17] ConfigMaps. (2023, 30 abril). Kubernetes. Visitat el 01/04/2023. URL: https://kubernetes.io/docs/concepts/configuration/configmap/

[18] Team, K. (2023, 16 març). ¿Qué es Ingress Controller en Kubernetes? KeepCoding Bootcamps. Visitat el 01/04/2023. URL: https://keepcoding.io/blog/que-es-ingress-controller-en-kubernetes/

[19] ¿Qué es Java? - Explicación del lenguaje de programación Java - AWS. (s. f.). Amazon Web Services, Inc. Visitat el 22/04/2023. URL: https://aws.amazon.com/es/what-is/java/

[20] Spring. (s. f.). Spring. Visitat el 22/04/2023. URL: https://spring.io/

[21] Qué es Framework. (s. f.). Arimetrics. Visitat el 22/04/2023. URL: https://www.arimetrics.com/glosario-digital/framework

[22] PostgreSQL. (s. f). PostgreSQL. Visitat el 07/05/2023. URL: https://www.postgresql.org/

[23] Postman. (s. f.). Visitat el 15/05/2023. URL: https://www.postman.com/

[24] Horizontal Pod Autoscaling. (2023, 30 març). Kubernetes. Visitat el 01/06/2023. URL: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

[25] Docker: Accelerated, Containerized Application Development. (2023, 22 febrero). Docker. Visitat el 12/03/2023. URL: https://www.docker.com

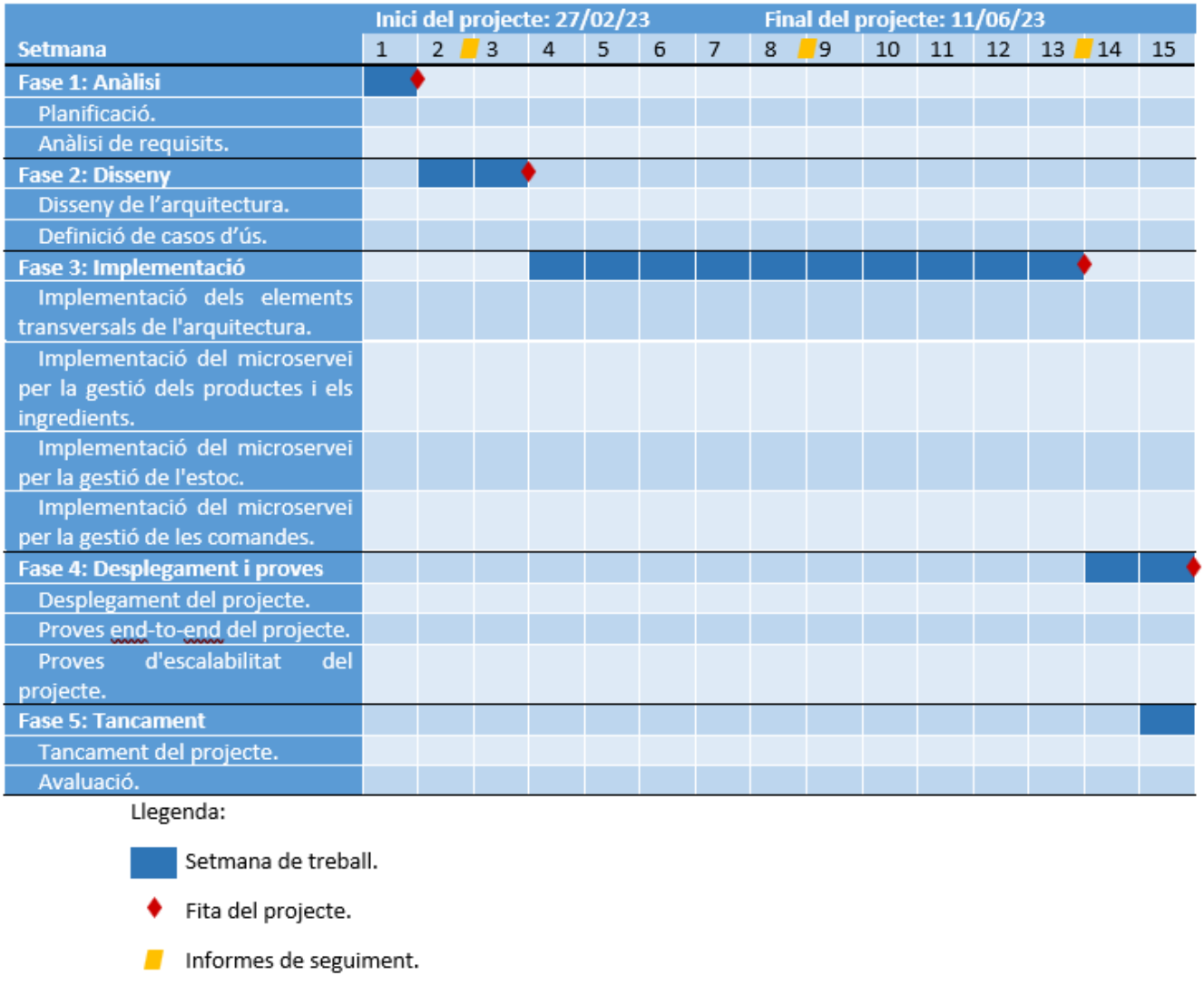
APÈNDIX

A1. REQUISITS DEL PROJECTE

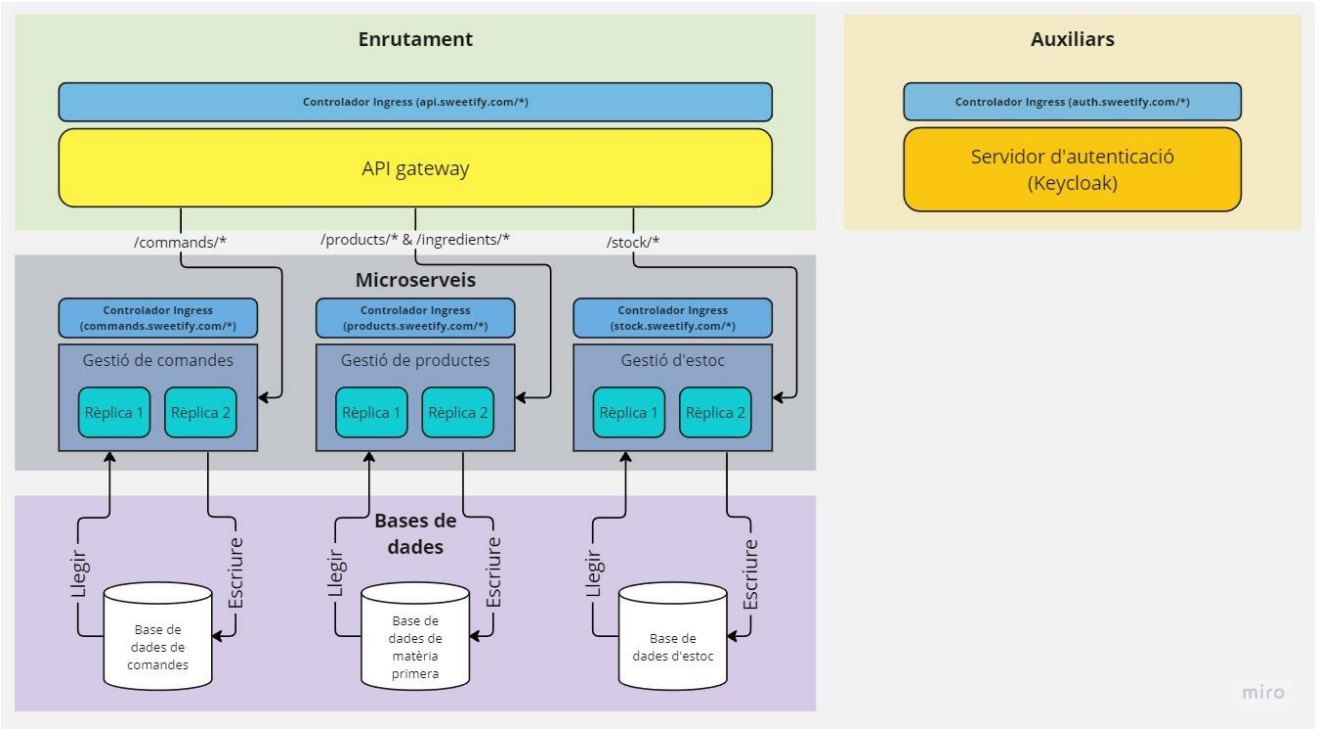
ID	Requisit funcionals	Prioritat
RF01	El sistema Sweetify ha de proveir a l'usuari la capacitat de veure un llistat de totes les comandes.	Alta
RF02	El sistema Sweetify ha de proveir a l'usuari la capacitat de veure les dades d'una comanda.	Mitja
RF03	El sistema Sweetify ha de proveir a l'usuari la capacitat de crear una comanda.	Alta
RF04	Tan aviat com es creï una comanda, el sistema Sweetify ha de passar la comanda a l'estat Per fer.	Alta
RF05	Tan aviat com es creï una comanda, el sistema Sweetify ha de restar l'estoc d'ingredients que necessita aquesta comanda.	Alta
RF06	El sistema Sweetify ha de proveir a l'usuari la capacitat de modificar una comanda.	Mitja
RF07	El sistema Sweetify ha de proveir a l'usuari la capacitat d'eliminar una comanda.	Alta
RF08	Tan aviat com s'elimini una comanda, si la comanda es troba en l'estat Per fer, el sistema Sweetify ha de sumar l'estoc d'ingredients que necessita aquesta comanda.	Alta
RF09	Si la comanda es troba en l'estat Per fer, el sistema Sweetify ha de proveir a l'usuari la capacitat de passar una comanda a l'estat En progrés.	Alta
RF10	Si la comanda es troba en l'estat En progrés, el sistema Sweetify ha de proveir a l'usuari la capacitat de passar una comanda a l'estat Preparada.	Alta
RF11	Si la comanda es troba en l'estat Preparada, el sistema Sweetify ha de proveir a l'usuari la capacitat de passar una comanda a l'estat Entregada.	Alta
RF12	El sistema Sweetify ha de proveir a l'usuari la capacitat de passar una comanda a l'estat Cancel·lada.	Alta
RF13	Tan aviat com es cancel·li una comanda, si la comanda es troba en l'estat Per fer, el sistema Sweetify ha de sumar l'estoc d'ingredients que necessita aquesta comanda.	Alta
RF14	Si la comanda es troba en l'estat Entregada o Cancel·lada, el sistema Sweetify ha de proveir a l'usuari la capacitat de passar una comanda a l'estat Per fer.	Alta
RF15	El sistema Sweetify ha de proveir a l'usuari la capacitat de veure un llistat de tots els productes.	Alta
RF16	El sistema Sweetify ha de proveir a l'usuari la capacitat de veure les dades d'un producte.	Mitja
RF17	El sistema Sweetify ha de proveir a l'usuari la capacitat de crear un producte.	Alta
RF18	El sistema Sweetify ha de proveir a l'usuari la capacitat de modificar un producte.	Mitja
RF19	Si el producte no té comandes assignades, el sistema Sweetify ha de proveir a l'usuari la capacitat d'eliminar un producte.	Baixa
RF20	El sistema Sweetify ha de proveir a l'usuari la capacitat d'afegir productes.	Alta
RF21	El sistema Sweetify ha de proveir a l'usuari la capacitat de consumir productes.	Alta
RF22	El sistema Sweetify ha de proveir a l'usuari la capacitat de veure un llistat de tots els ingredients.	Alta
RF23	El sistema Sweetify ha de proveir a l'usuari la capacitat de veure les dades d'un ingredient.	Mitja
RF24	El sistema Sweetify ha de proveir a l'usuari la capacitat de crear un ingredient.	Alta
RF25	El sistema Sweetify ha de proveir a l'usuari la capacitat de modificar un ingredient.	Mitja
RF26	Si l'ingredient no té productes assignats, el sistema Sweetify ha de proveir a l'usuari la capacitat d'eliminar un ingredient.	
RF27	El sistema Sweetify ha de proveir a l'usuari la capacitat d'afegir estoc a un ingredient.	Alta
RF28	El sistema Sweetify ha de proveir a l'usuari la capacitat de treure estoc a un ingredient.	Alta
RF29	El sistema Sweetify ha de proveir a l'usuari la capacitat de veure un llistat de tots els estocs.	Alta
RF30	El sistema Sweetify ha de proveir a l'usuari la capacitat de veure les dades d'un estoc.	Mitja
RF31	El sistema Sweetify ha de proveir a l'usuari la capacitat de crear un estoc.	Alta
RF32	El sistema Sweetify ha de proveir a l'usuari la capacitat de modificar un estoc.	Mitja
RF33	El sistema Sweetify ha de proveir a l'usuari la capacitat d'eliminar un estoc.	Baixa
RF34	Tan aviat com la data de caducitat d'un estoc hagi passat, el sistema Sweetify podrà eliminar l'estoc caducat.	Baixa

ID	Requisits no funcionals	Prioritat
RNF01	El sistema ha de tenir una disponibilitat del 99%.	Alta
RNF02	Cada microservei del sistema ha de tenir disponibles un mínim de dues rèpliques.	Alta
RNF03	El sistema ha de donar d'alta una nova rèplica d'un microservei quan el percentatge d'ús de la CPU estigui entorn el 50% fins a un límit de 5 rèpliques.	Alta
RNF04	En cas que una rèplica falli, el sistema ha de donar d'alta un altre en menys de 2 minuts.	Mitja
RNF05	El sistema ha de ser capaç de processar amb normalitat fins a un volum de 50 peticions concurrents.	Baixa

A2. Diagrama de Gantt del projecte



A3. DIAGRAMA DE COMPONENTS DE L'ARQUITECTURA



A4. DIAGRAMA DEL FLUX DE PETICIONS