



Máster en Cloud Apps
Desarrollo y despliegue de aplicaciones en la nube

Curso académico 2019/2020

Trabajo de Fin de Máster

RUP y Arquitecturas A-giles

Autores:

Carlos Corcobado Vaz

Neofitos Koutsourais Blázquez

Francisco Arboleya García

Tutor: Luis Fernández

Table of Contents

Agradecimientos.....	6
Resumen	7
Introducción	8
Objetivo	8
Modelo del dominio	9
Requisitos	10
Actores y casos de uso.....	10
Contexto de casos de uso	11
Especificaciones de casos de uso	12
CreateUser.....	12
ActivateUser	13
DeactivateUser	13
UserLogin.....	14
UserLogout	14
RecoveryPassword.....	15
SetNewPassword	15
ShowUserProfile	16
CreateProduct.....	17
ShowProductDetail.....	18
UnsharedProduct.....	18
AccessBorrowPlace.....	19
SearchInBorrowPlace.....	19
CreateBorrow	20
ConfirmBorrow	20
AddCommentToBorrow	21
RejectBorrow	21
SetScoreBorrow	22
ShowBorrowList.....	23
ShowBorrowDetail.....	23
ShowUserScore.....	24
ShowProductList.....	24
UpdateUserProfile	25
EditBorrow.....	26
DeleteProduct.....	26

EditProduct	27
Prototipos de interfaces	28
CreateUser	28
ActivateUser	29
DeactivateUser	29
UserLogin	30
UserLogout	30
RecoveryPassword	31
SetNewPassword	31
ShowUserProfile	32
CreateProduct	33
ShowProductDetail	34
UnsharedProduct	34
AccessBorrowPlace	35
SearchInBorrowPlace	35
CreateBorrow	36
ConfirmBorrow	36
AddCommentToBorrow	37
RejectBorrow	37
SetScoreBorrow	38
ShowBorrowList	38
ShowBorrowDetail	39
ShowUserScore	39
ShowProductList	40
UpdateUserProfile	40
EditBorrow	41
DeleteProduct	41
EditProduct	42
Análisis	43
CreateUser	43
ActivateUser	43
DeactivateUser	44
UserLogin	44
UserLogout	45
RecoveryPassword	45
SetNewPassword	46

ShowUserProfile	46
CreateProduct	47
ShowProductDetail	47
UnsharedProduct	48
AccessBorrowPlace	48
SearchInBorrowPlace	49
CreateBorrow	49
ConfirmBorrow	50
AddCommentToBorrow	50
RejectBorrow	51
SetScoreBorrow	51
ShowBorrowList	52
ShowBorrowDetail	52
ShowUserScore	53
ShowProductList	53
UpdateUserProfile	54
EditBorrow	54
DeleteProduct	55
EditProduct	55
Diseño	56
Diseño de la arquitectura	56
Capa de datos	59
Diseño de casos de uso	61
Implementación	62
Ecosistema de desarrollo	62
Calidad del software	63
Fiabilidad	65
Mantenibilidad	65
Seguridad	66
Tamaño	66
Duplicidades	67
Complejidad	68
Cobertura de código	69
Pruebas	70
Evaluación de las pruebas	71
Despliegue	72

Control de versiones.....	72
Entorno Cloud.....	74
Despliegue	76
Diagrama de Hardware.....	77
Entorno de desarrollo.....	78
Arquitecturas A-giles y tal y tal	79
Objetivos.....	79
Definiciones	80
Beneficios y desventajas.....	83
Resumen	85
Conclusiones.....	86
Bibliografía.....	87

Agradecimientos

A nuestras familias por aguantarnos con el master desde el curso 2019 – 2021, hasta hoy abril 2021 en el que entregamos el TFM.

- Patricia y Zoe
- Ana y Carmen
- Maria y Vega

“al lado de unos pobres hombres siempre hay unas grandes mujeres”

Al equipo de profesores del Máster Cloud Apps, con los que hemos compartido mucho tiempo y una cuarentana en remoto inédita hasta el momento para todos nosotros.

Y en especial a Luis Fernandez, por aguantarnos mutuamente y habernos ayudado a poner los pies en la tierra, enseñarnos a comunicarnos y entendernos con un lenguaje único. Y fundamentalmente a ver el mundo como un gran MVC.

Resumen

El objeto de este documento es afianzar y aunar los conocimientos que hemos ido adquiriendo en cada una de las asignaturas del máster.

Para ello hemos diseñado e implementado una solución enfocada a un producto final de préstamo de productos entre personas. Utilizando para ello la metodología Rational Unified Process, la solución corre bajo .net Core 5.0 permitiendo así el despliegue multiplataforma. Las tecnologías empleadas han sido C# para la implementación del BackEnd, Bootstrap, Razor y jQuery para el FrontEnd, así como Entity Framework Core como ORM para la interacción con la BBDD alojada en SQL Server.

Además, nos hemos apoyado en Azure como proveedor cloud, y herramientas como GitHub como control de código fuente, GitHub Actions para la orquestación del CI/CD, Sengrid como servicio de email marketing, Azure Blob Storage.

En el primer capítulo se introduce brevemente el alcance del proyecto y se expone la definición del modelo del dominio.

A lo largo del segundo punto haremos una descripción detallada de los requisitos del sistema definidos a través de los casos de uso.

Durante el tercer y cuarto punto se explica el análisis y diseño del sistema. Con la representación conceptual a través de los diagramas de colaboración.

En el quinto punto se trata lo referente a la implementación, así como las herramientas y tecnologías usadas y el porqué de la elección de cada una de ellas.

En el sexto punto se explica la implementación de pruebas llevadas a cabo, así como una breves descripción de la problemática encontrada y posterior corrección.

En el séptimo punto se expone la solución elegida para el despliegue del proyecto.

En el octavo punto se hace una comparativa y valoración de las diferentes arquitecturas A-giles más actuales frente a MVC

Introducción

Objetivo

El objetivo de este proyecto es afianzar todo lo aprendido durante el curso y aunar todos los conceptos en un mínimo producto viable.

Para ello hemos aplicado los conceptos de las siguientes asignaturas,

- Diseño y calidad software
- Patrones y arquitectura software
- Pruebas software
- Seguridad por cultura
- Pruebas de servicios de Internet
- Arquitecturas de servicios de Internet
- Persistencia y análisis de datos
- Computación en la nube
- Contenedores y orquestadores
- DevOps
- Repositorios y modelos de desarrollo
- Integración y entrega continua
- Despliegue continuo

Nos hemos apoyado como metodología principal de desarrollo en Rational Unified Process. Usando como tecnología principal Net 5 y lenguaje mayoritario C#, además de otros como Javascript, Razor y Bootstrap y SQL Server apoyado sobre el ORM Entity Framework Core para la capa de persistencia.

Para proveer una red de seguridad a nuestro proyecto nos hemos apoyado en el framework xUnit para pruebas unitarias y funcionales.

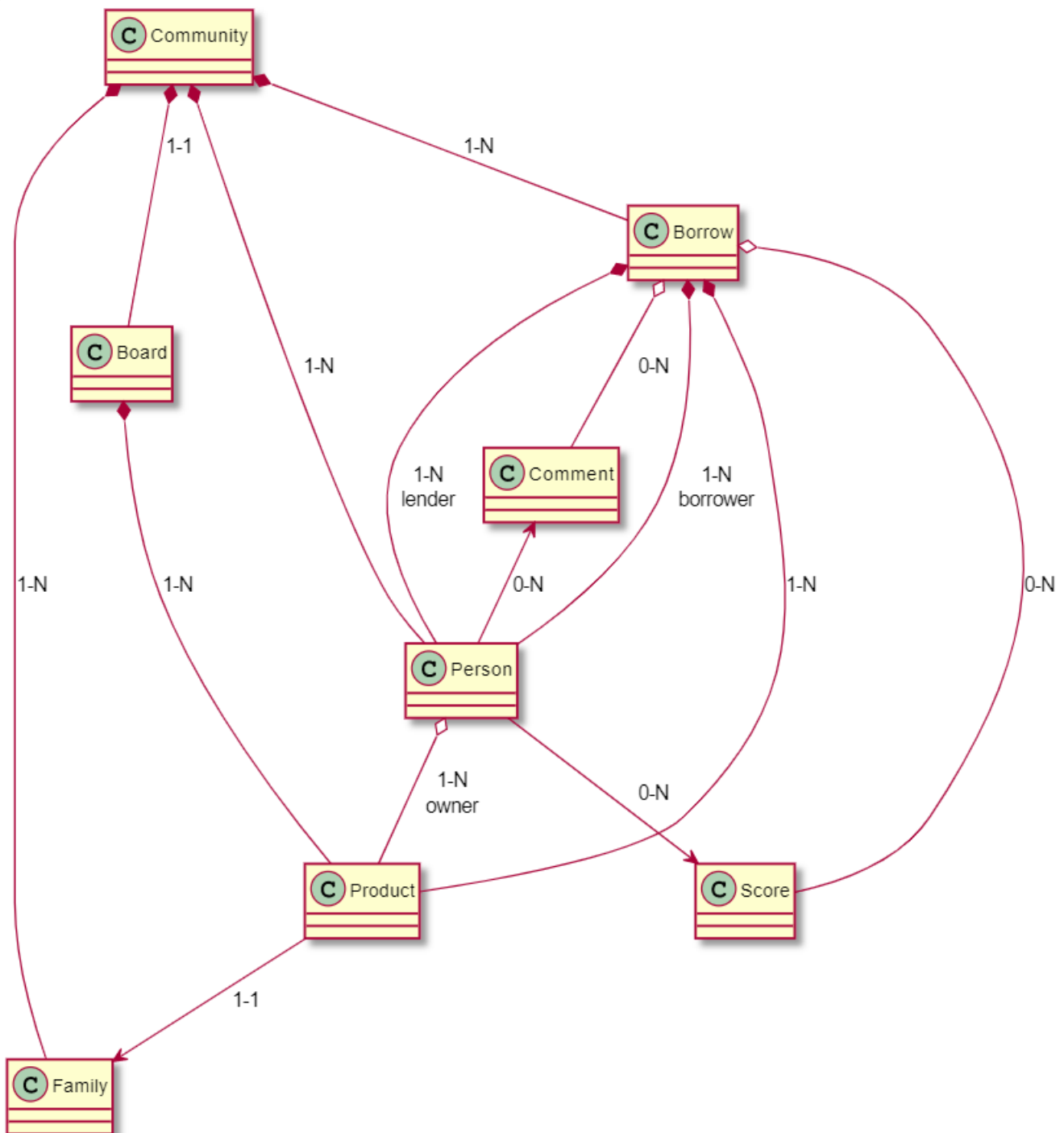
La aplicación web desarrollada se llama ShareThings. La cual consiste en dar a las personas una herramienta para compartir productos que no utilicen en un periodo de tiempo pactado, es decir, realizar prestamos de productos entre personas de forma desinteresada.

Modelo del dominio

A continuación, se presenta el diagrama del modelo del dominio, en el cual se describen los conceptos más importantes del contexto como objetos del dominio y las relaciones entre ellos.

- Objetos de negocio
- Objetos del mundo real
- Eventos de ocurrencia.

Con esto se pretende unificar el vocabulario y conceptos clave del problema.

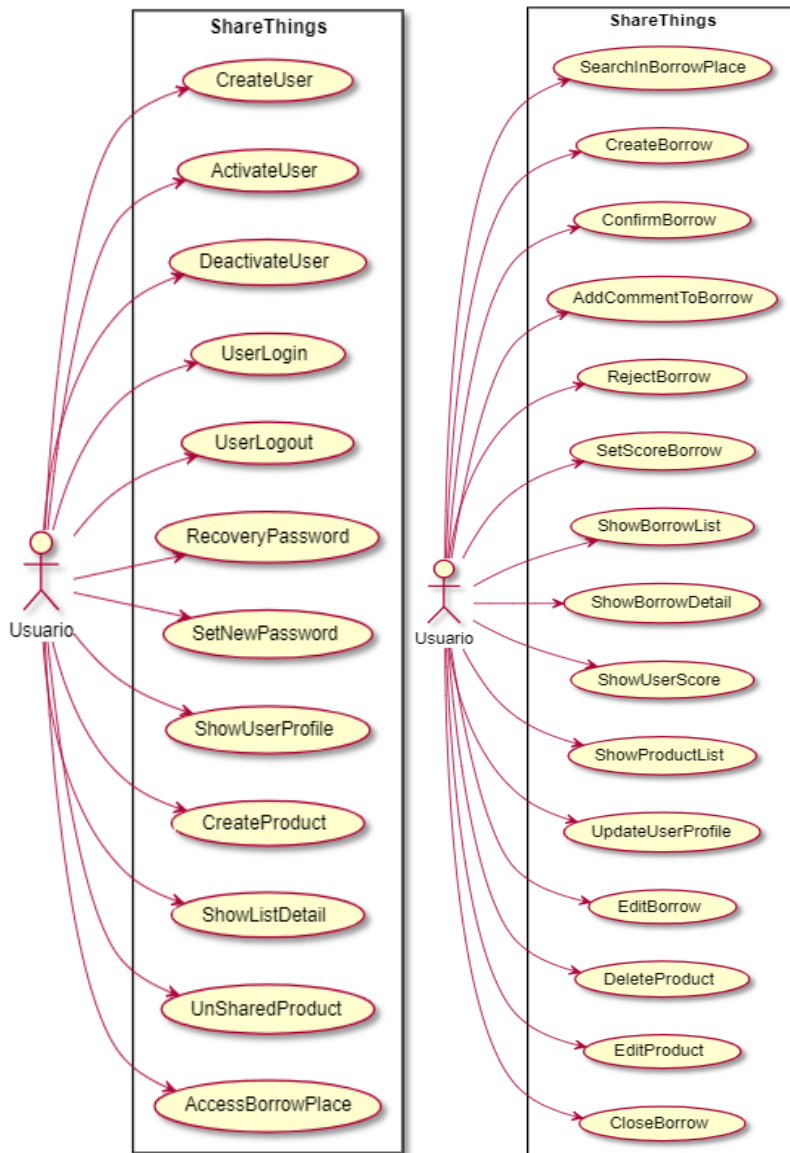


Requisitos

En este apartado se describen los requisitos del sistema, cuyo principal propósito es dirigir el desarrollo hacia el sistema correcto, de tal forma que se establezca y mantenga un acuerdo entre los desarrolladores, usuarios y clientes, se definan los límites del sistema, se provean las bases para planificar los contenidos técnicos de cada iteración, se provea a los desarrolladores del sistema de una mejor comprensión de los requisitos del sistema y se provean las bases para la estimación de costes y tiempos para desarrollar el sistema.

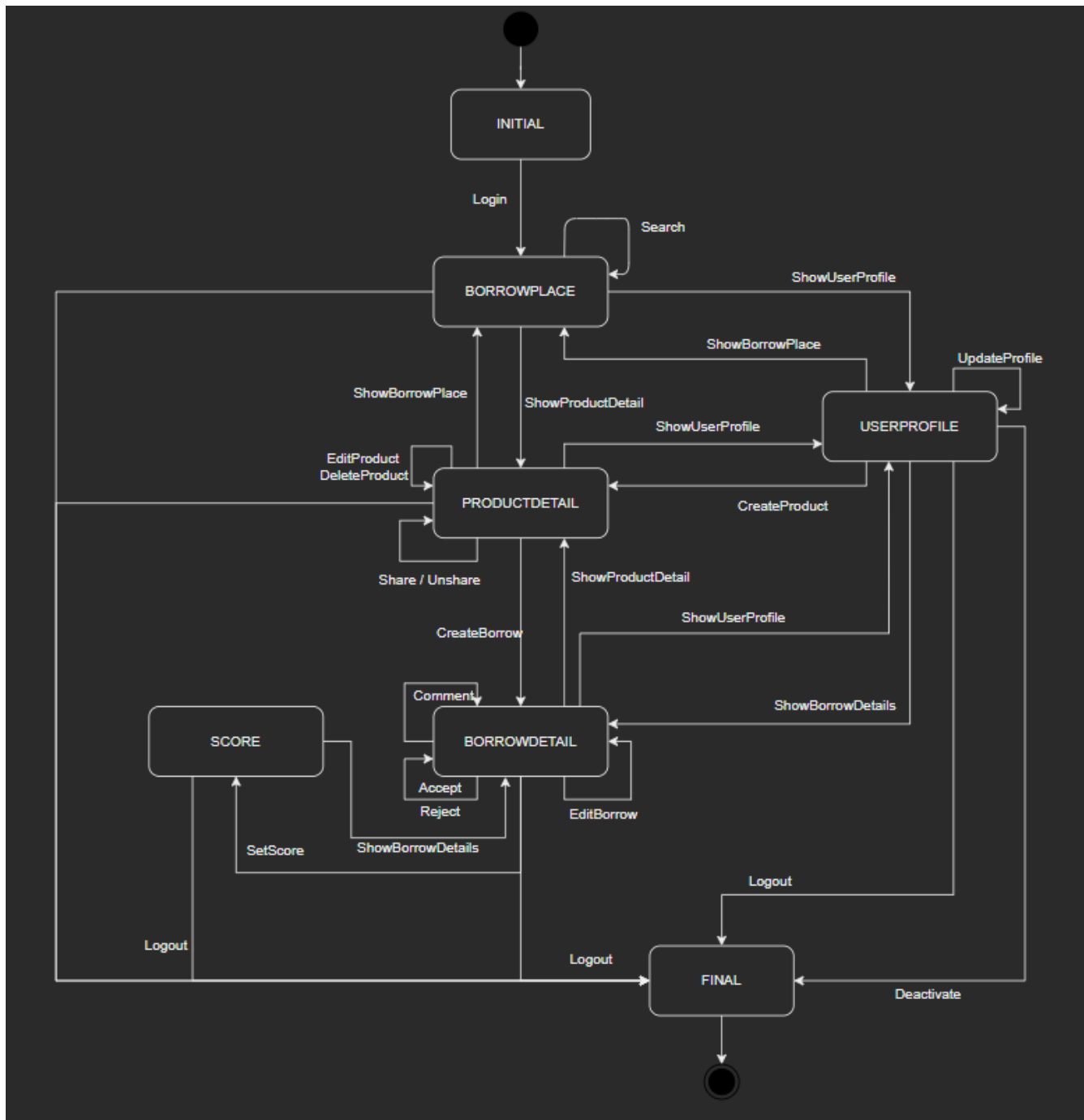
Actores y casos de uso

A continuación, se presenta el diagrama de casos de uso del sistema. En este diagrama se presenta el único actor de este sistema, el cual al interactuar con el sistema adopta el rol usuario, y veintisiete (27) casos de uso a realizar, los cuales son especificaciones de secuencias de interacciones que el sistema puede realizar.



Contexto de casos de uso

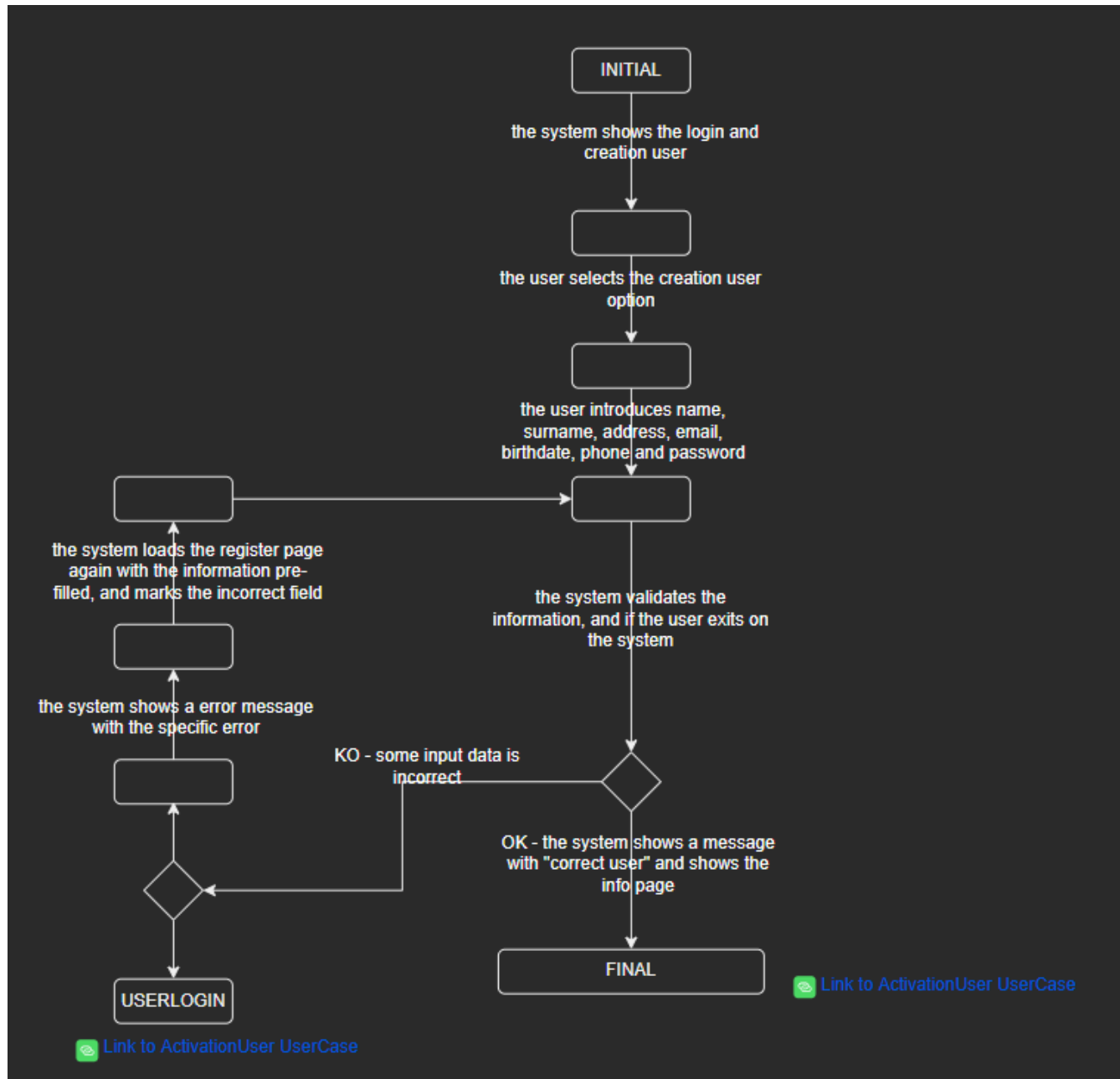
A continuación, se muestra el diagrama de contexto de casos de uso, mediante un diagrama de estados se representa el flujo de interacción entre los diferentes casos de uso que contempla el sistema. Cada recuadro representa un estado y cada flecha el caso de uso que conlleva a la transición de un estado a otro.



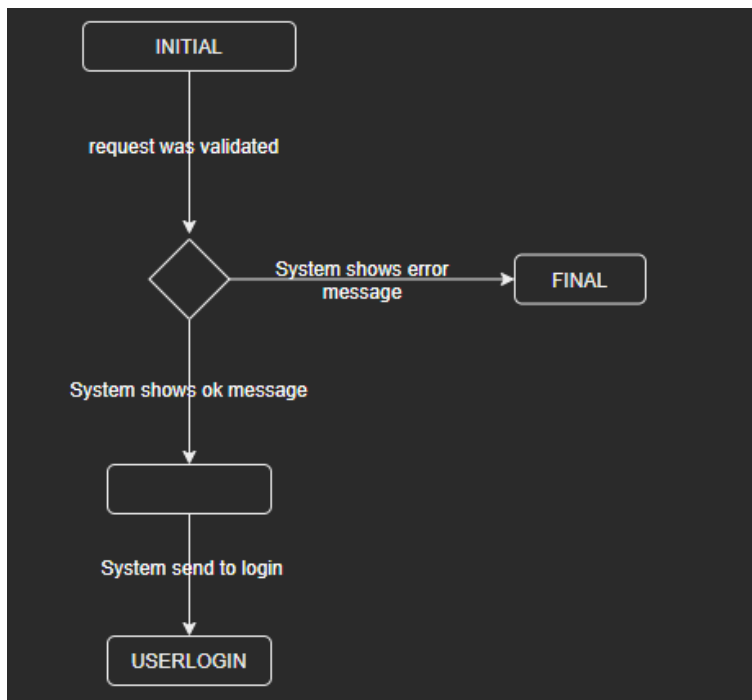
Especificaciones de casos de uso

En este apartado se describirán las secuencias de acciones de los casos de uso, cómo estas acciones son invocadas por los actores y como es el resultado de su ejecución a partir del actor, para ello se utilizarán diagramas de estado.

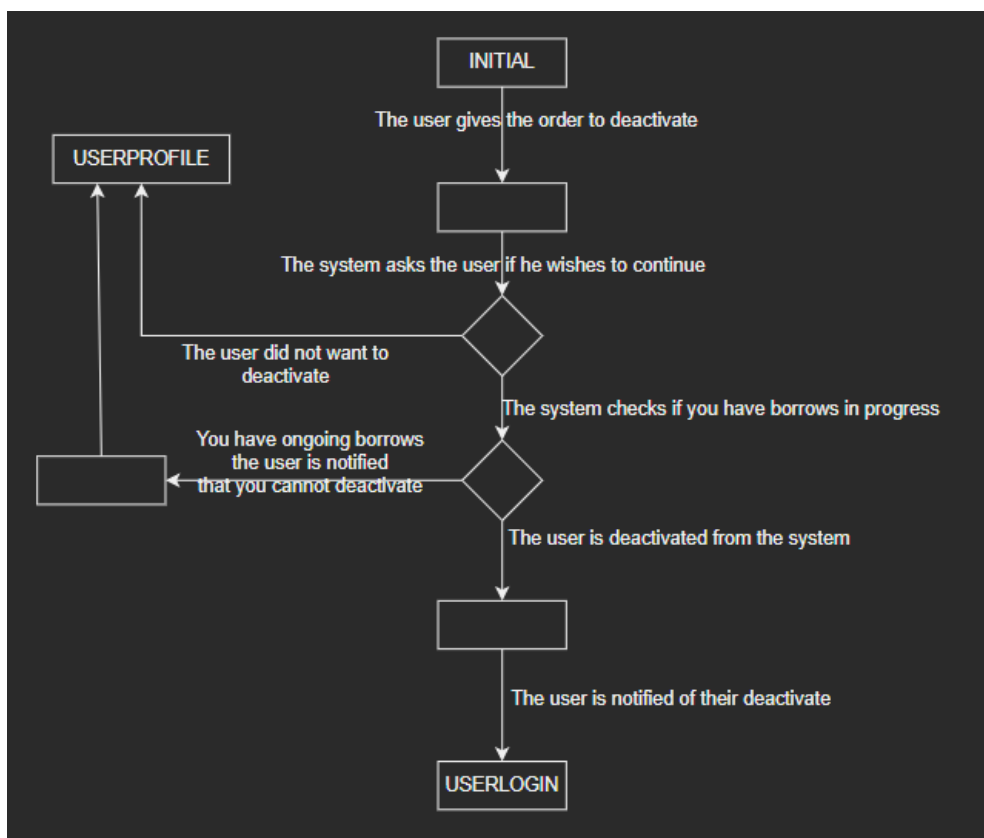
CreateUser



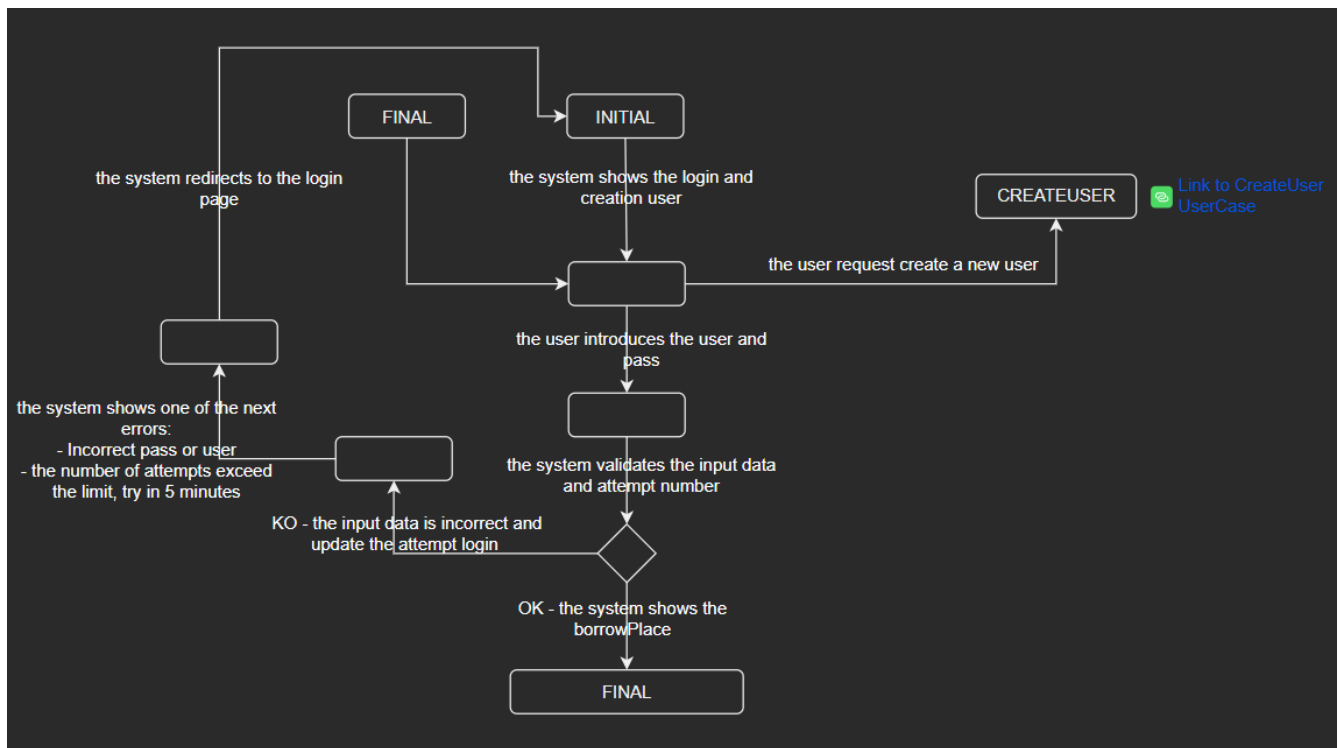
ActivateUser



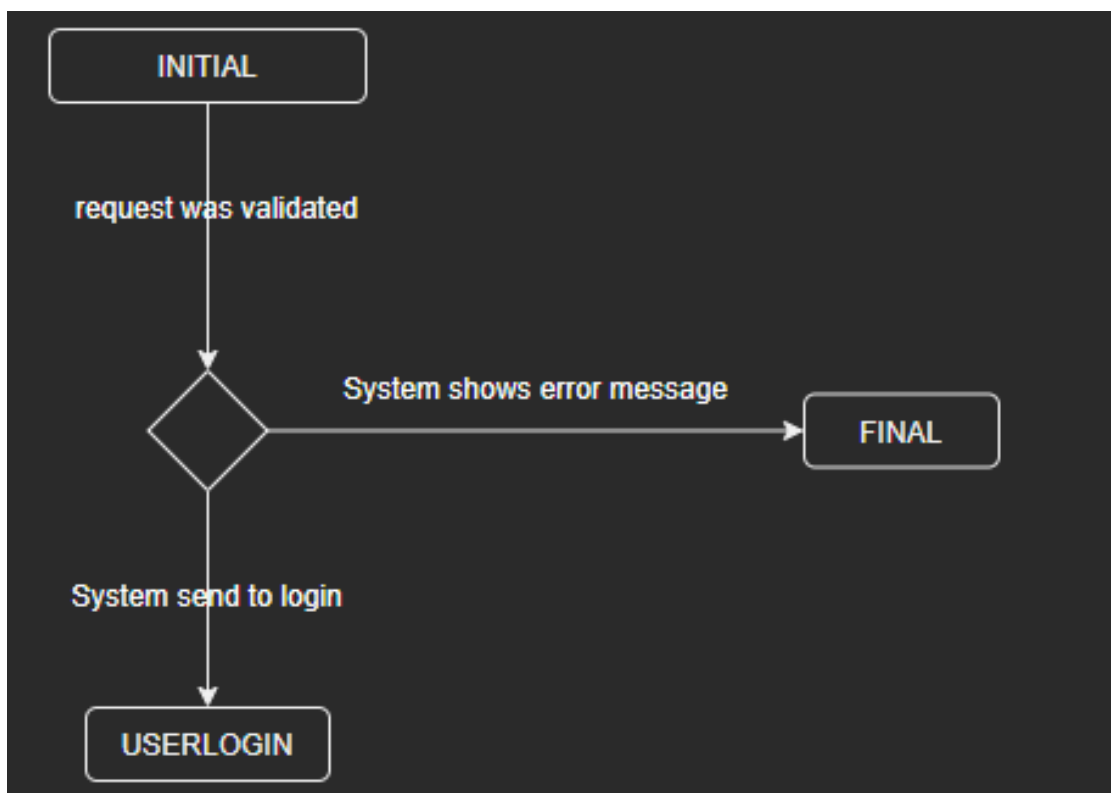
DeactivateUser



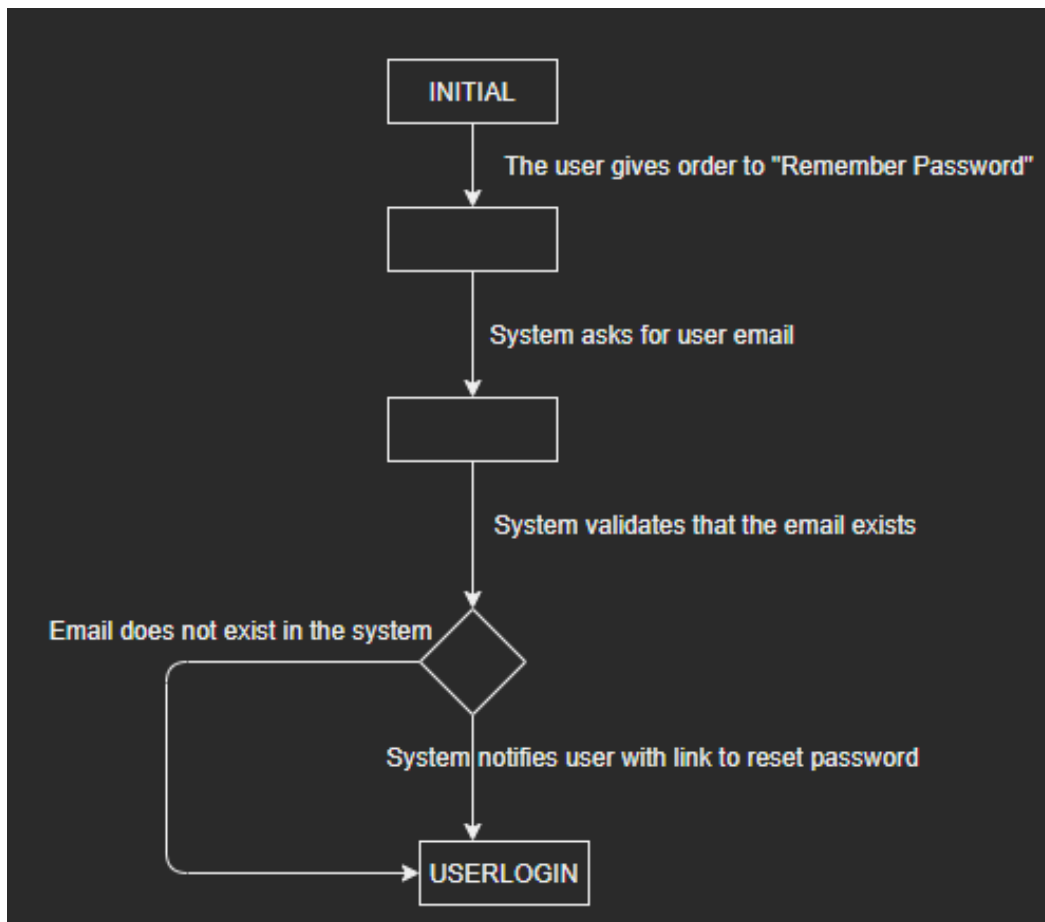
UserLogin



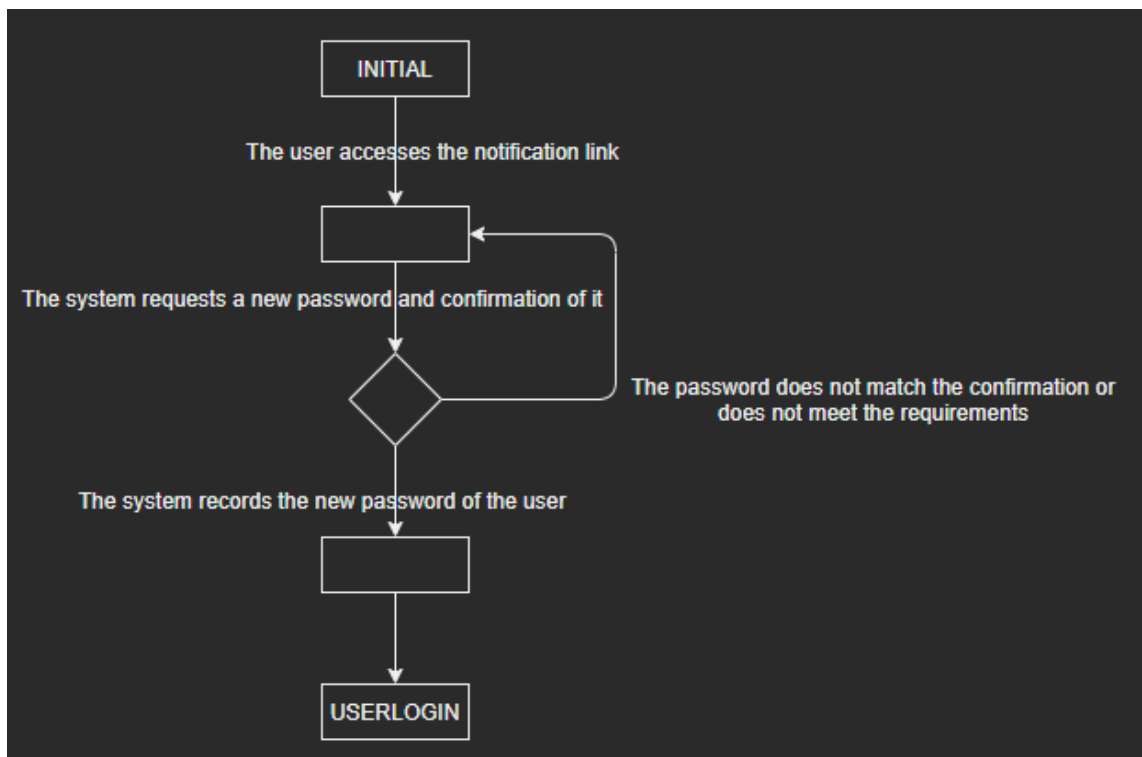
UserLogout



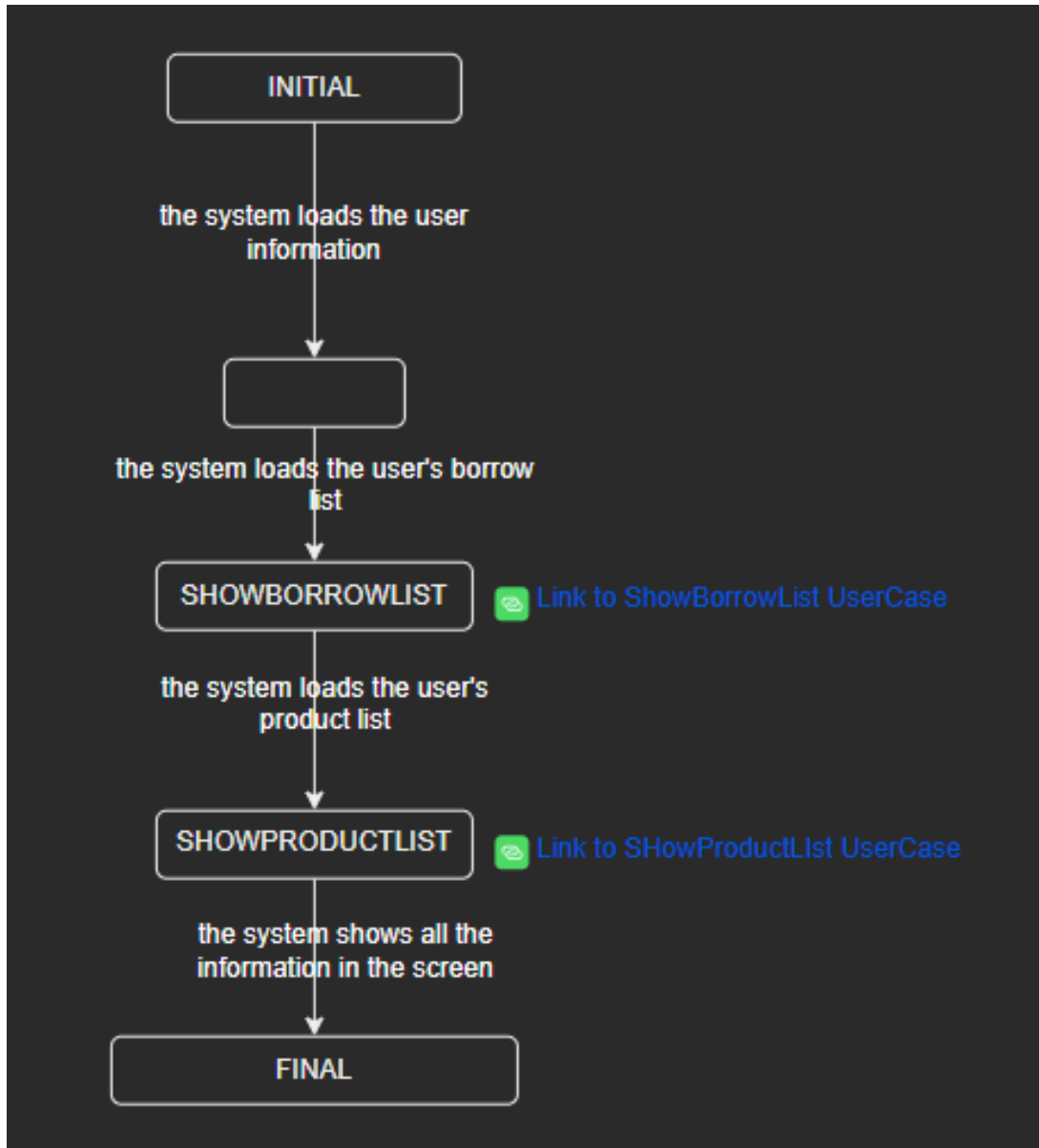
RecoveryPassword



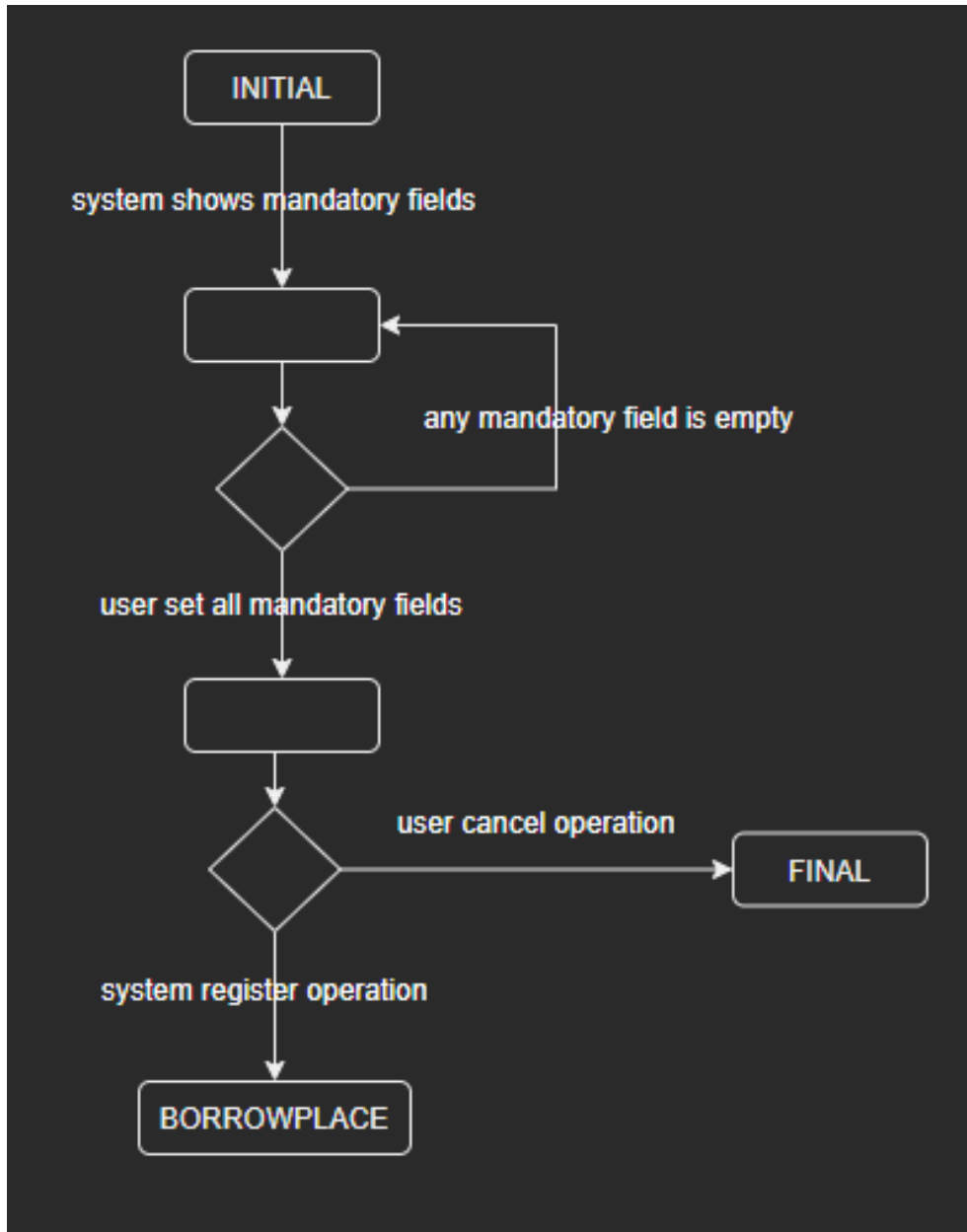
SetNewPassword



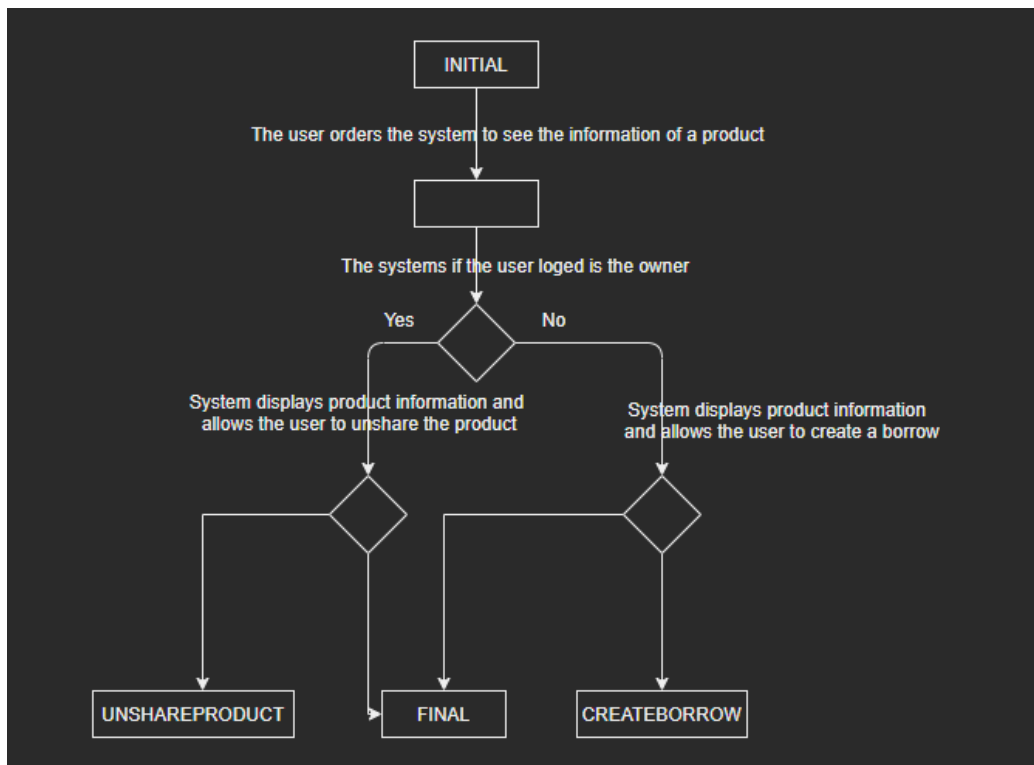
ShowUserProfile



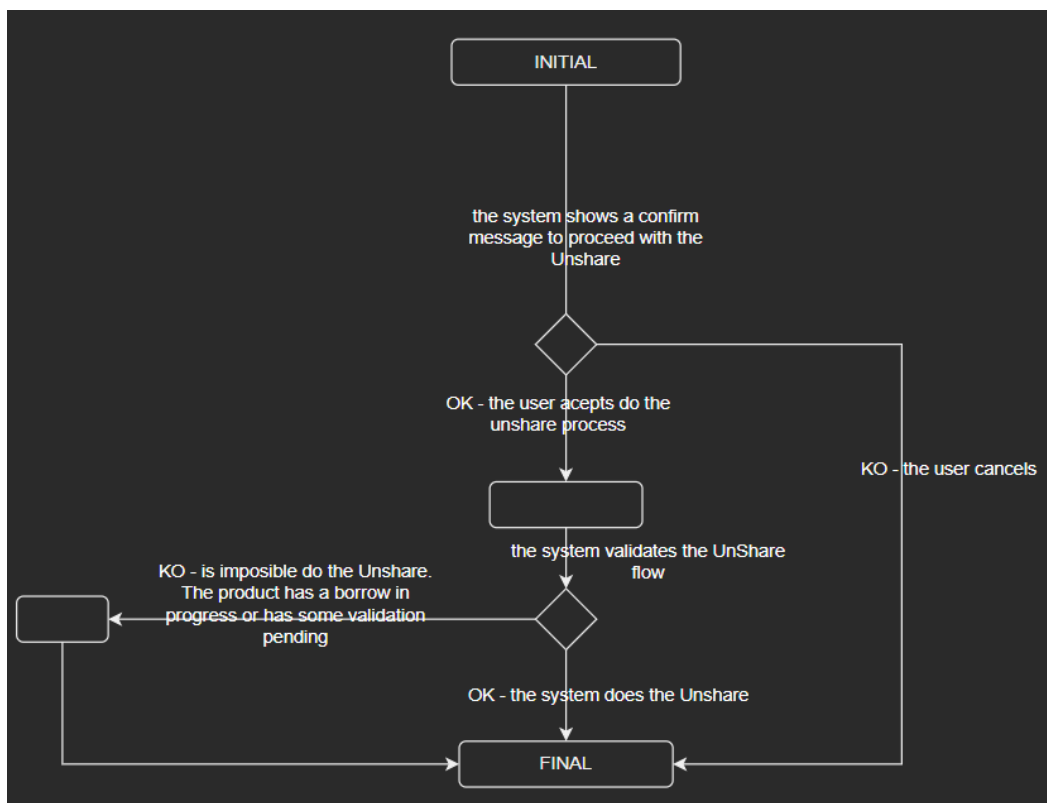
CreateProduct



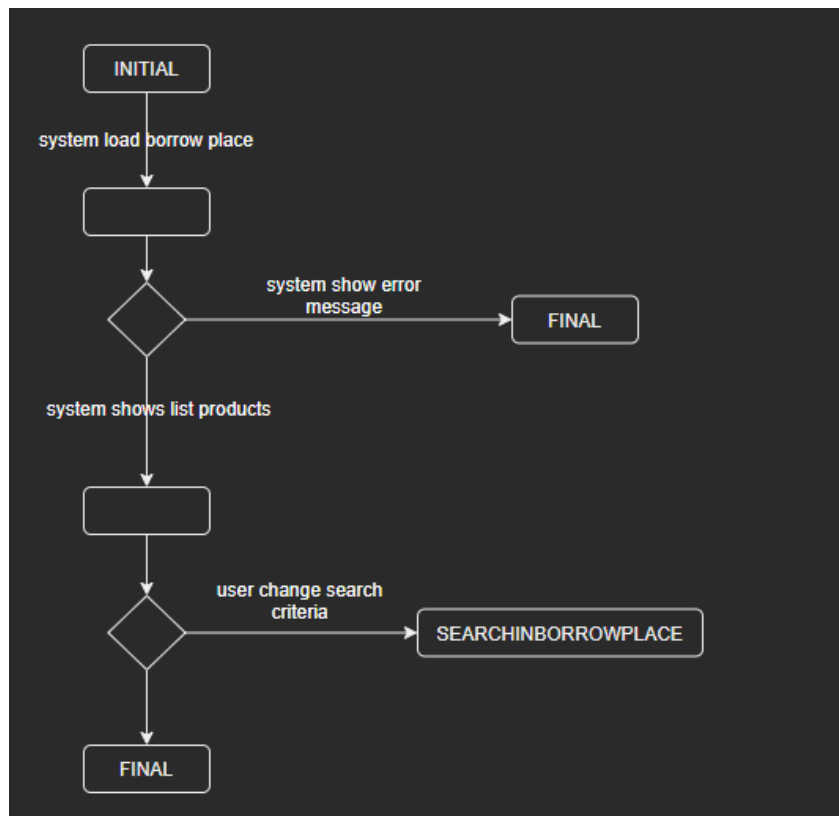
ShowProductDetail



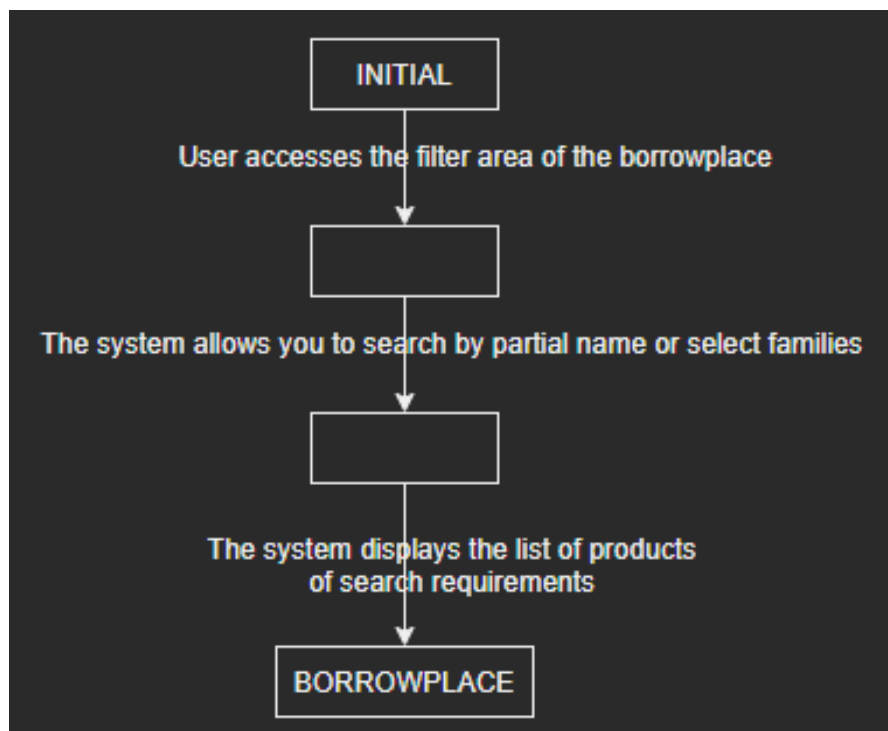
UnsharedProduct



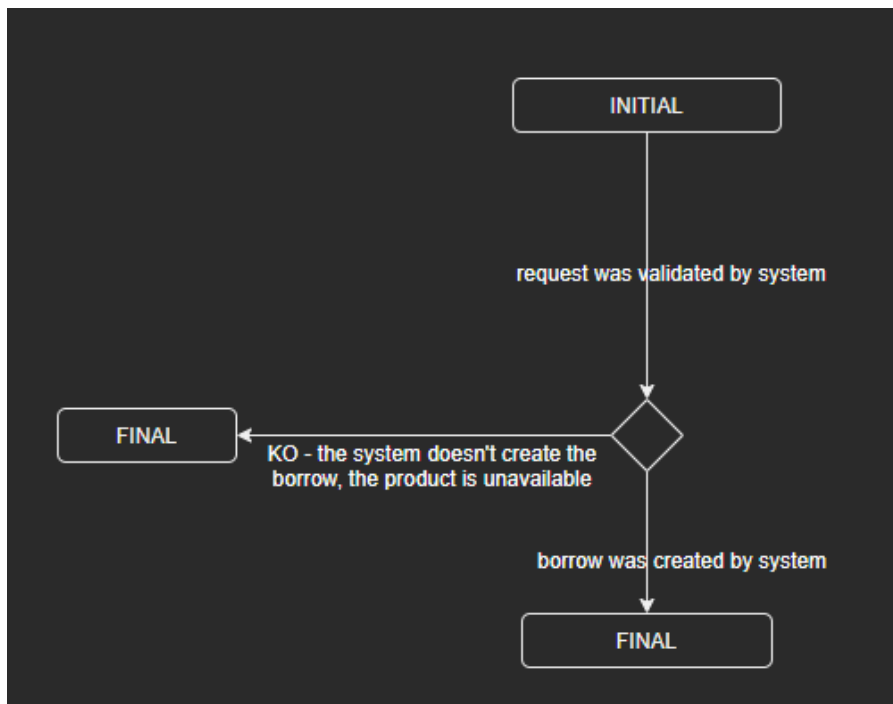
AccessBorrowPlace



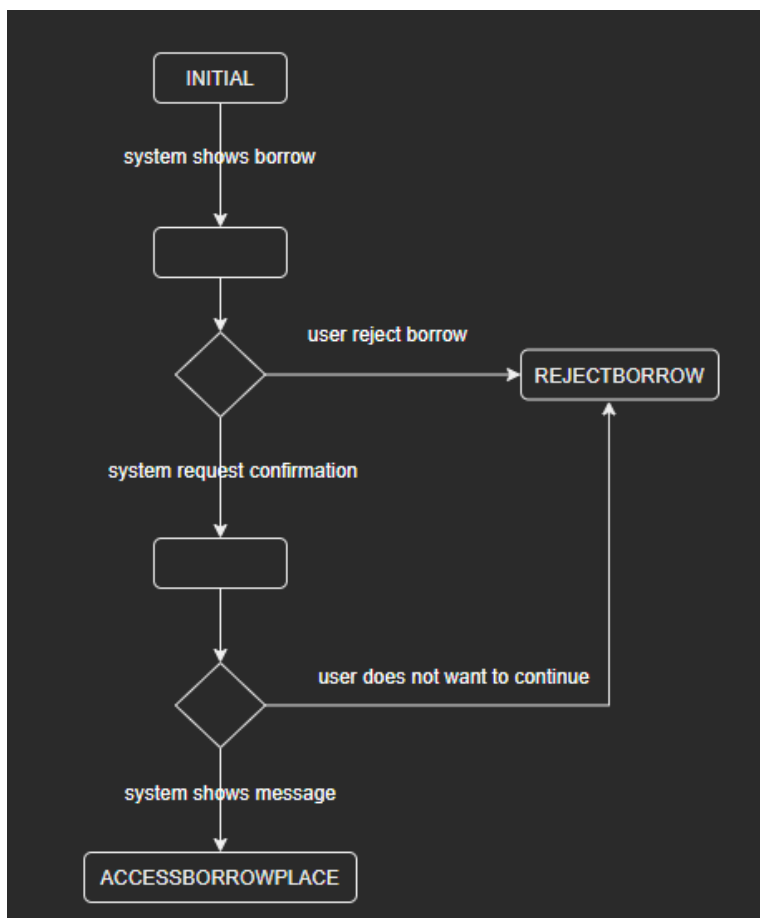
SearchInBorrowPlace



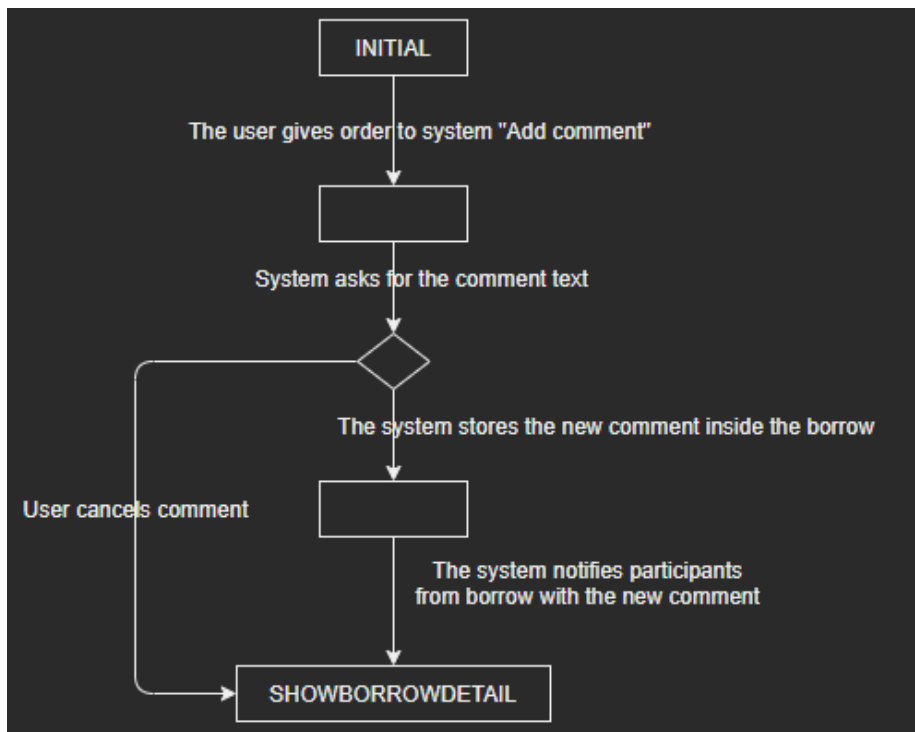
CreateBorrow



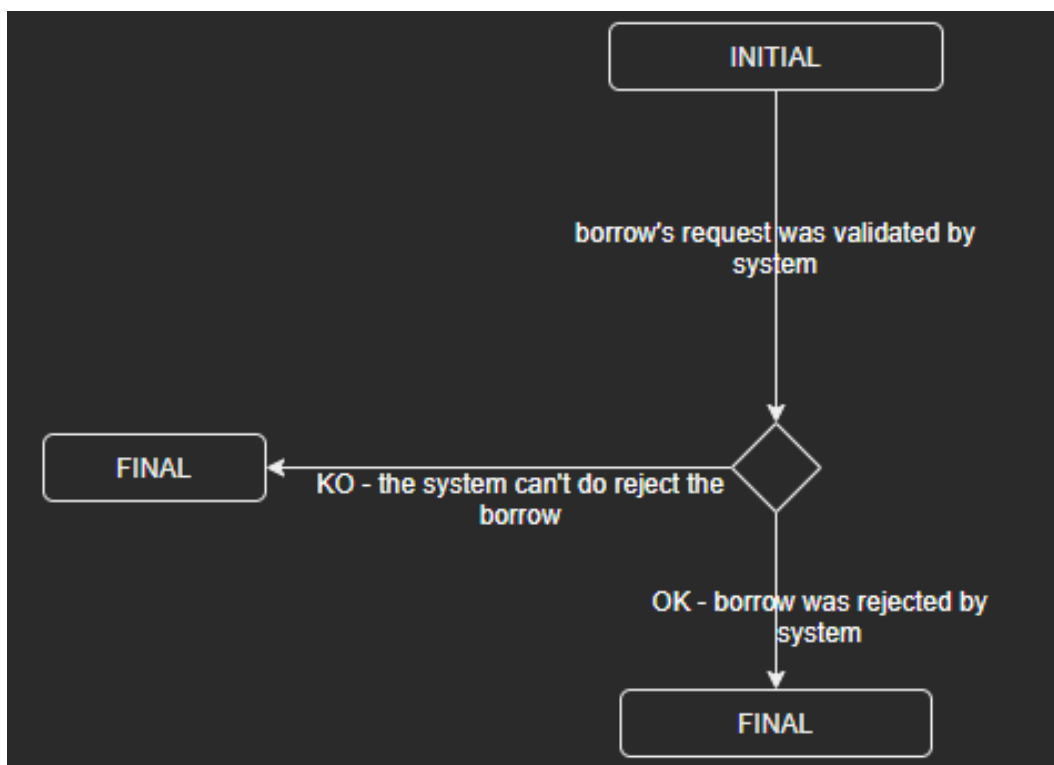
ConfirmBorrow



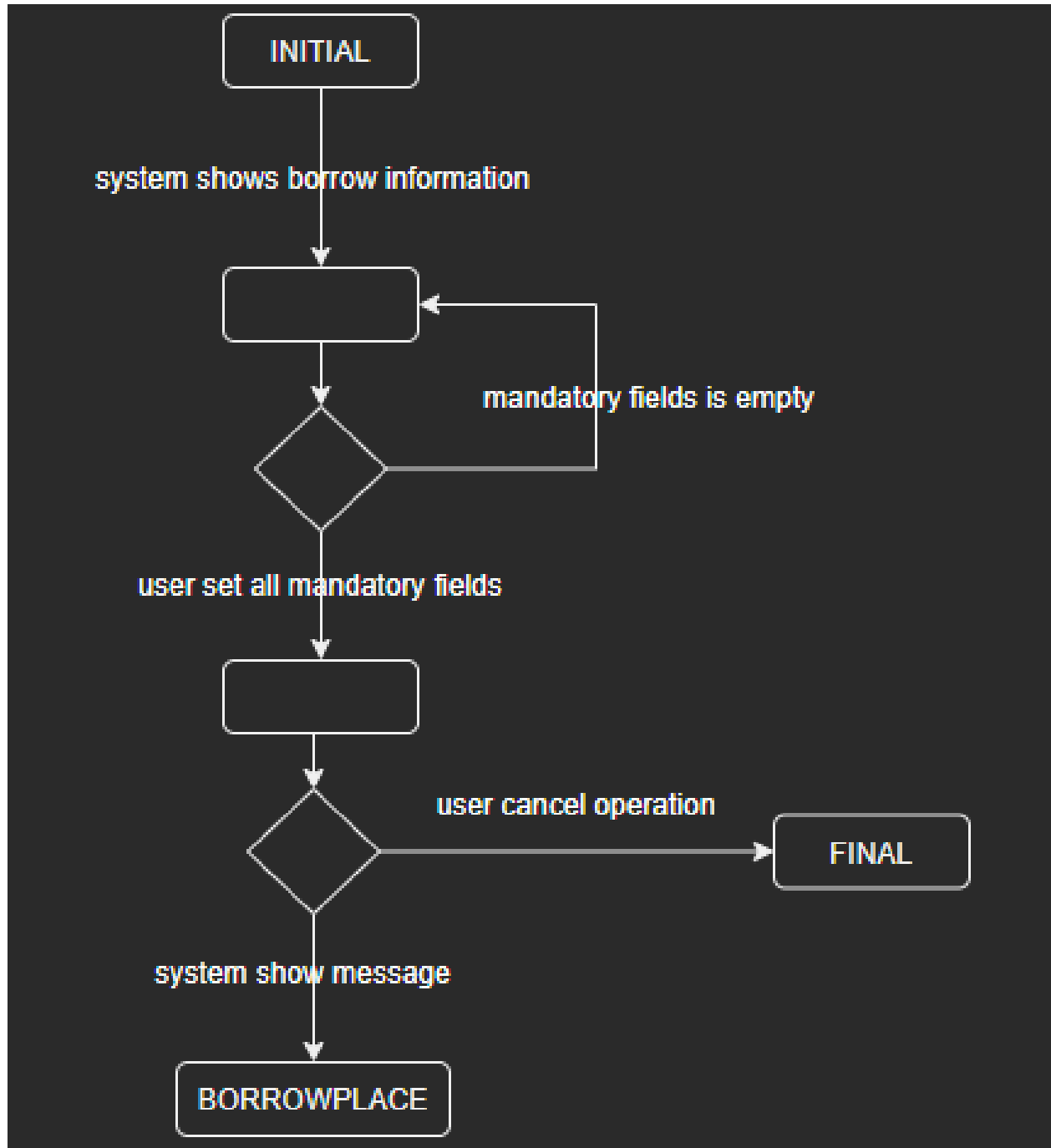
AddCommentToBorrow



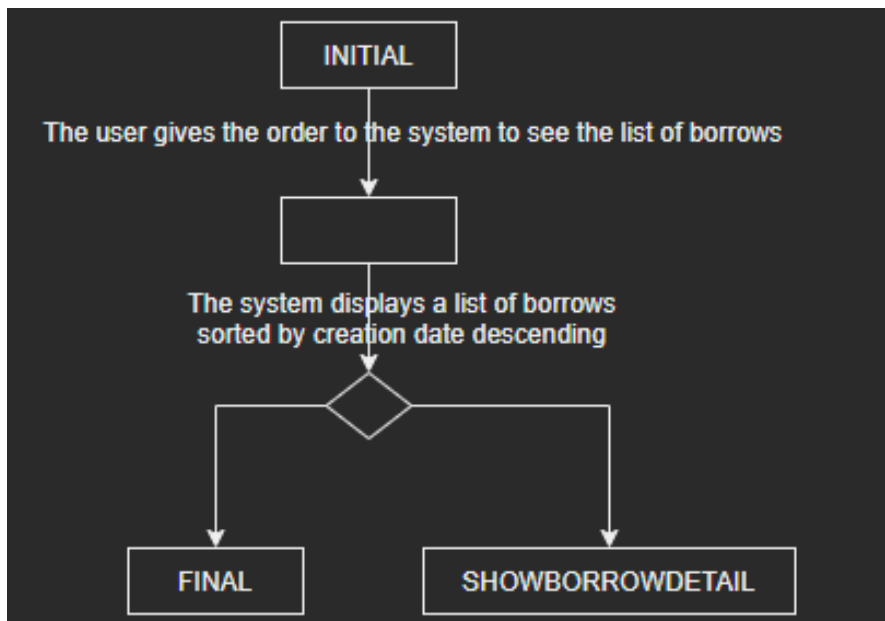
RejectBorrow



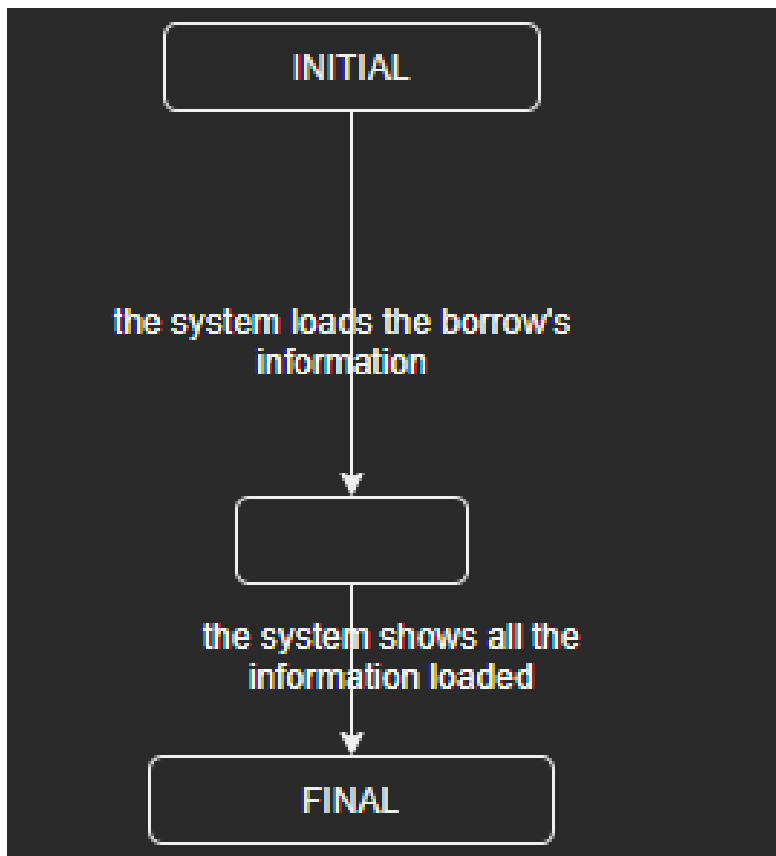
SetScoreBorrow



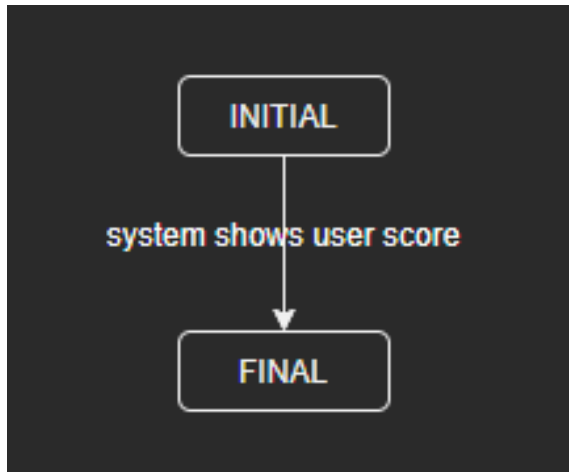
ShowBorrowList



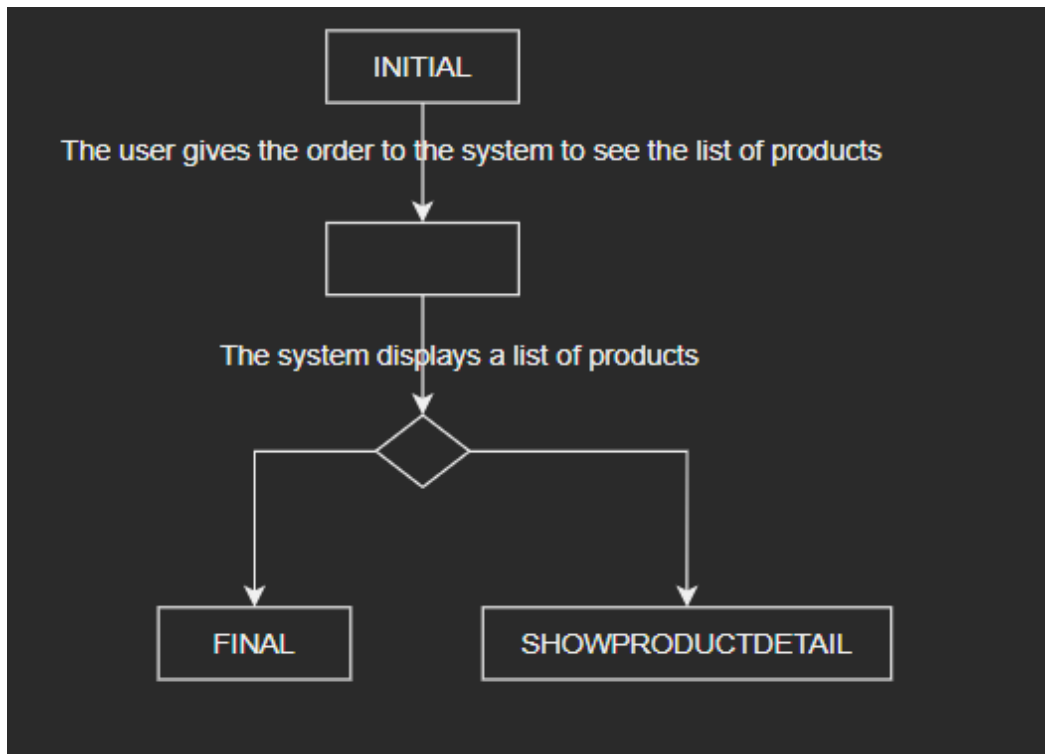
ShowBorrowDetail



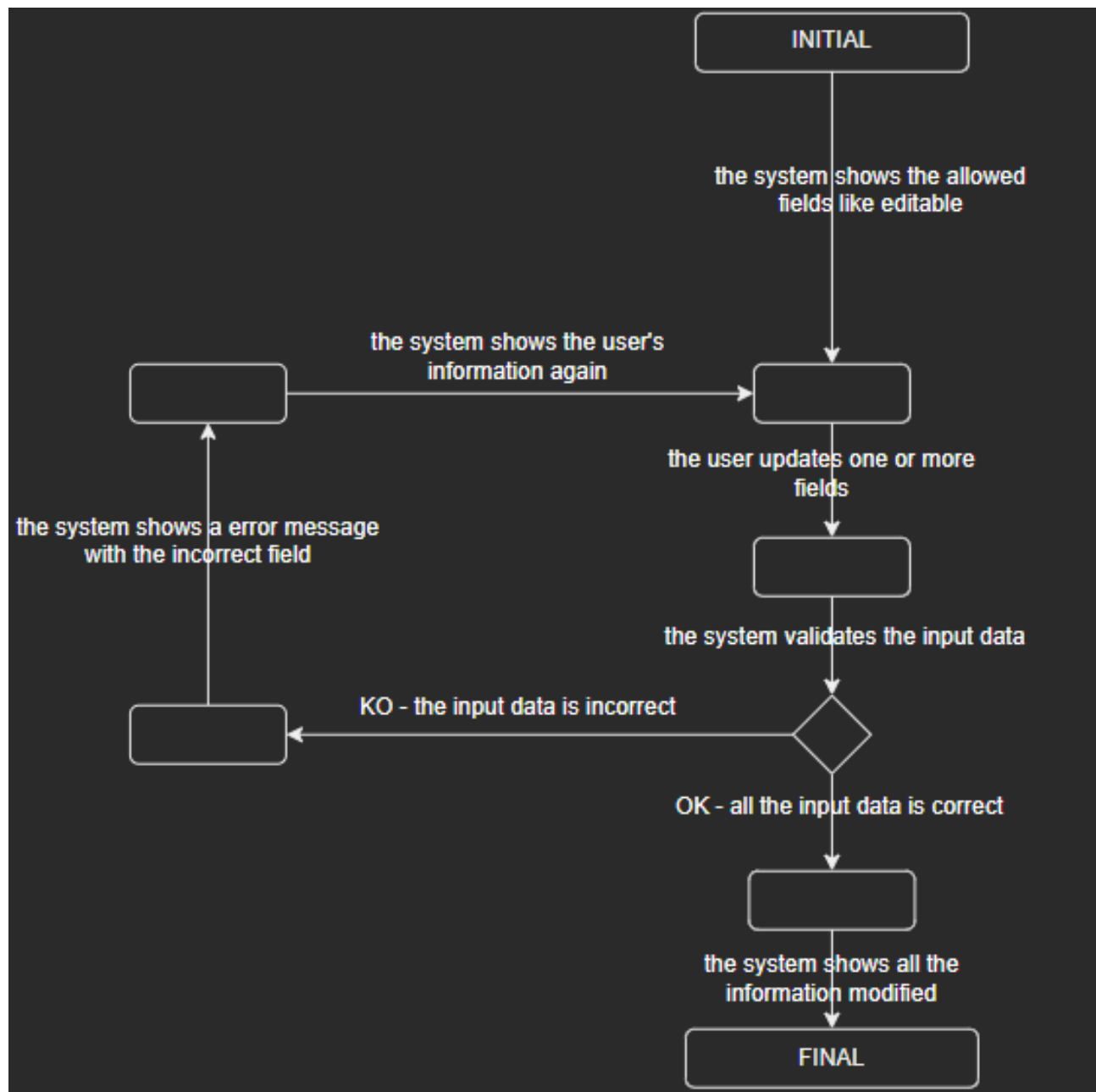
ShowUserScore



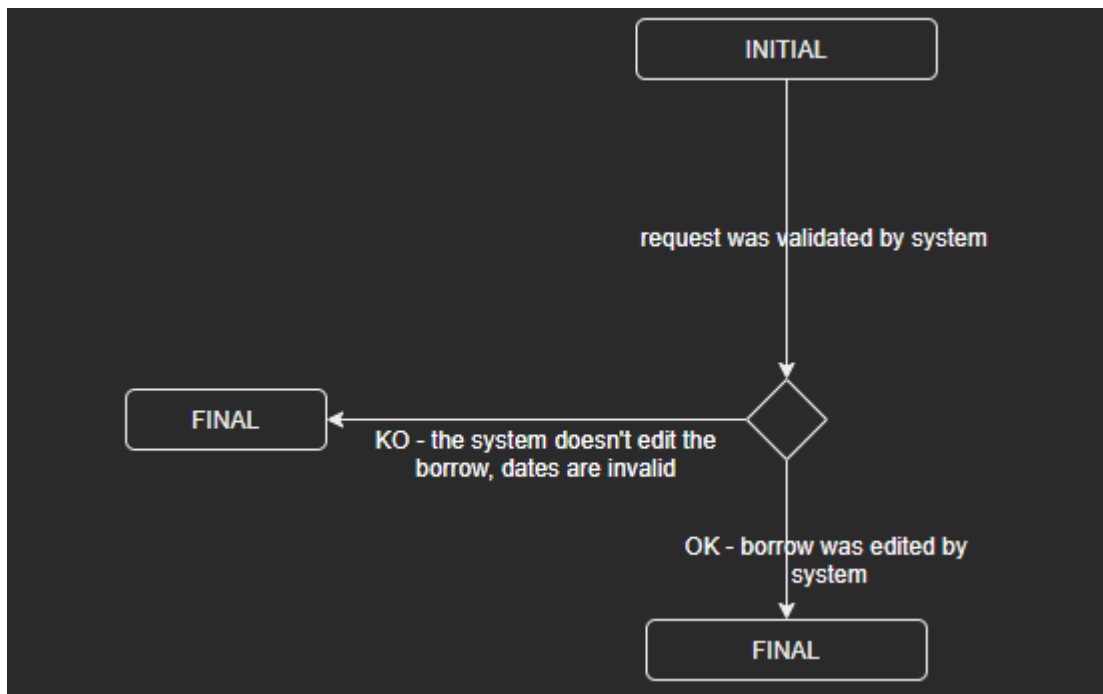
ShowProductList



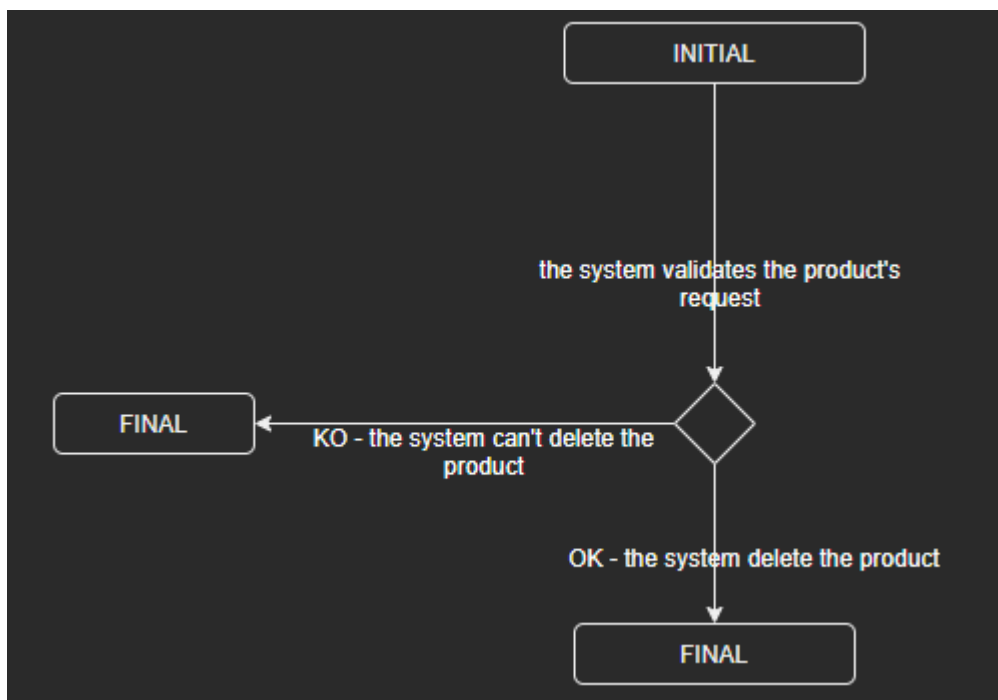
UpdateUserProfile



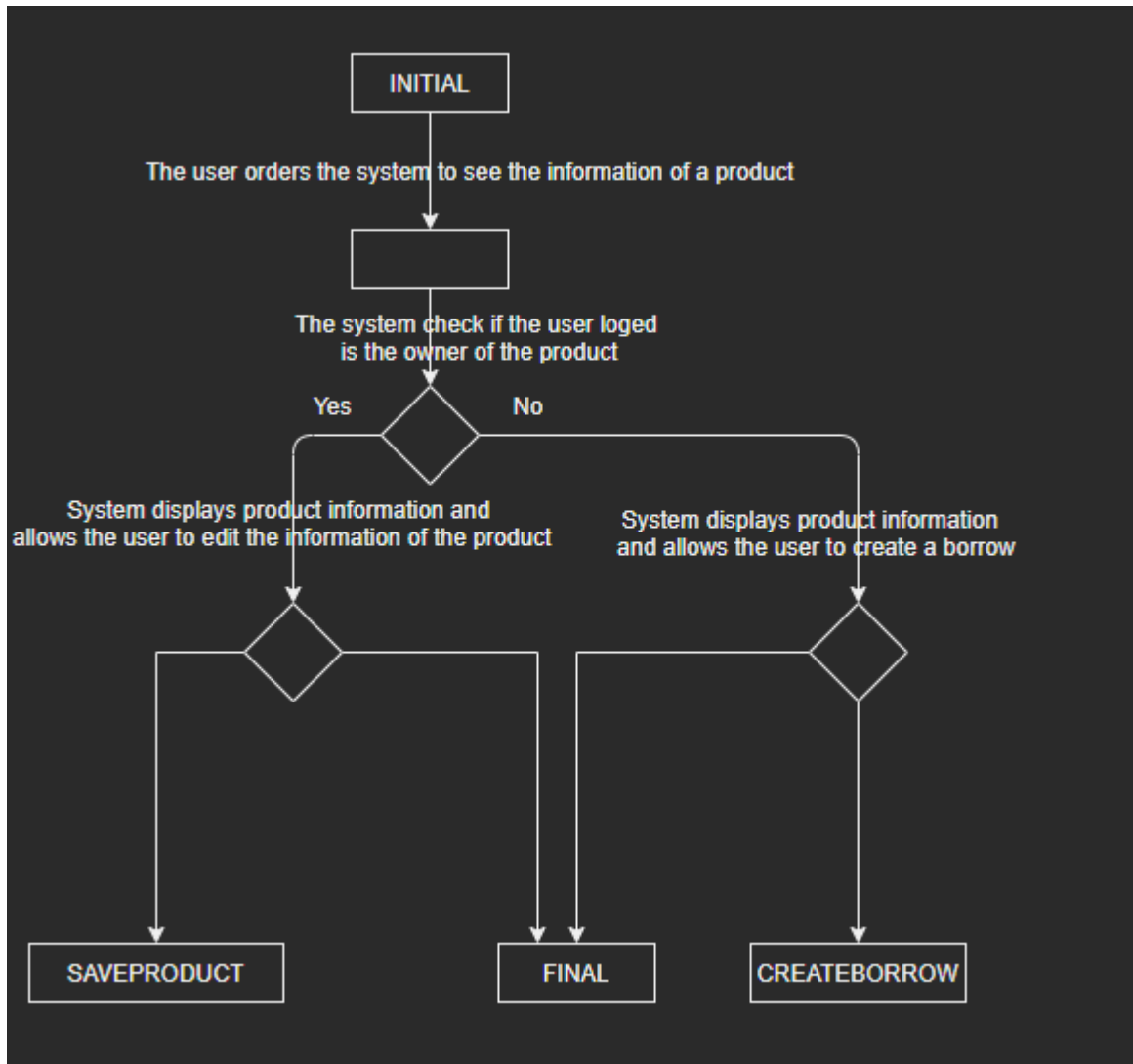
EditBorrow



DeleteProduct



EditProduct



Prototipos de interfaces

A continuación, se muestran los prototipos de interfaces que se ajustan a las historias de usuario. Estos han sido realizados con la herramienta Balsamiq.

CreateUser

SharyThings - User Creation


https://sharythings.com

SharyThings Platform

Name

Surname

Address

Birth Date 

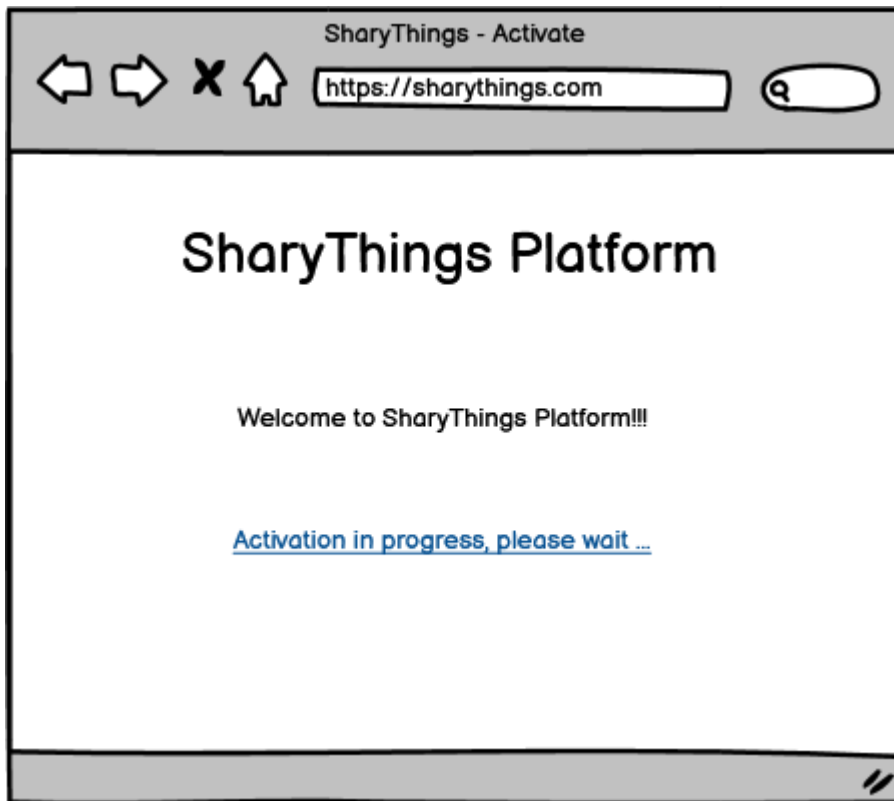
Phone

Mail

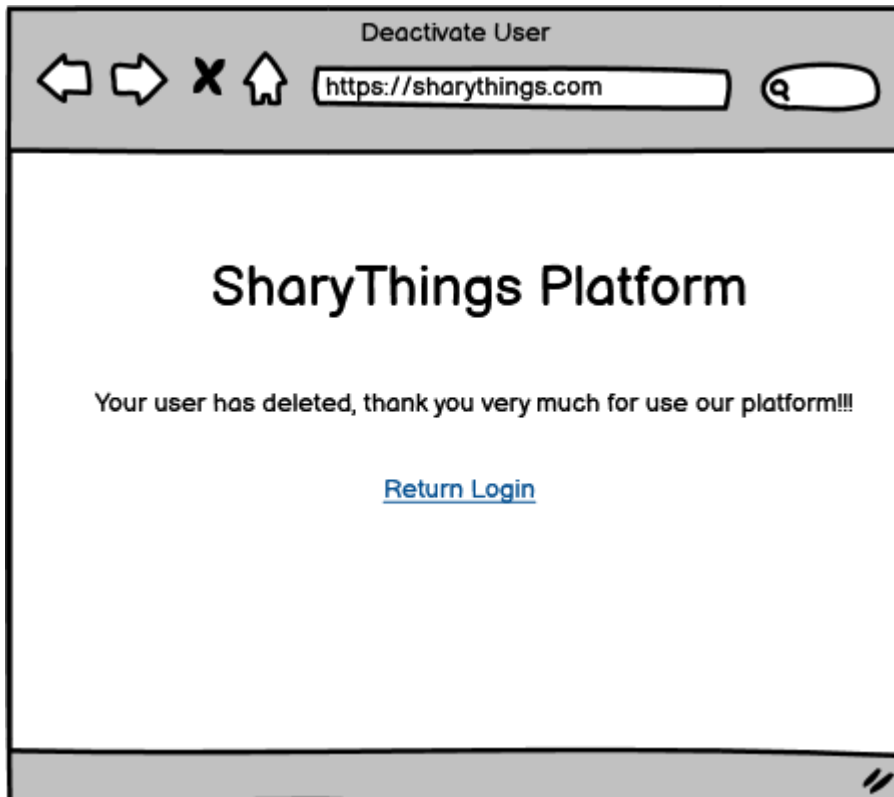
Password

Create

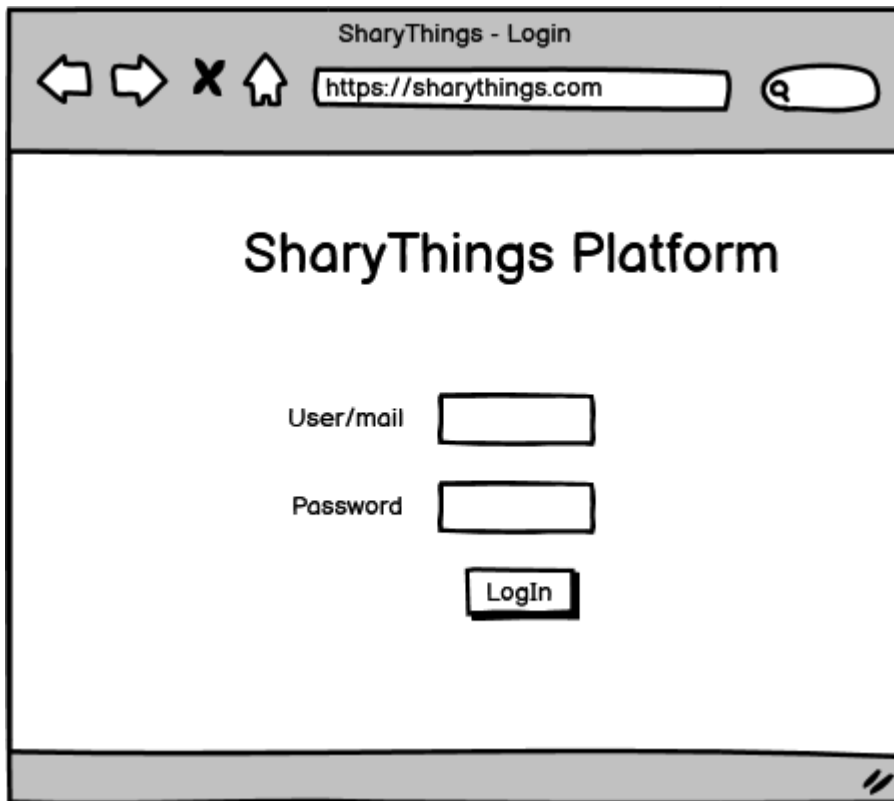
ActivateUser



DeactivateUser



UserLogin



A hand-drawn sketch of a web browser window titled "SharyThings - Login". The address bar shows "https://sharythings.com". The main content area displays "SharyThings Platform" in a large font. Below this, there are two input fields: "User/mail" and "Password". A "LogIn" button is positioned below the "Password" field. The browser window has a grey header and footer bar.

SharyThings - Login

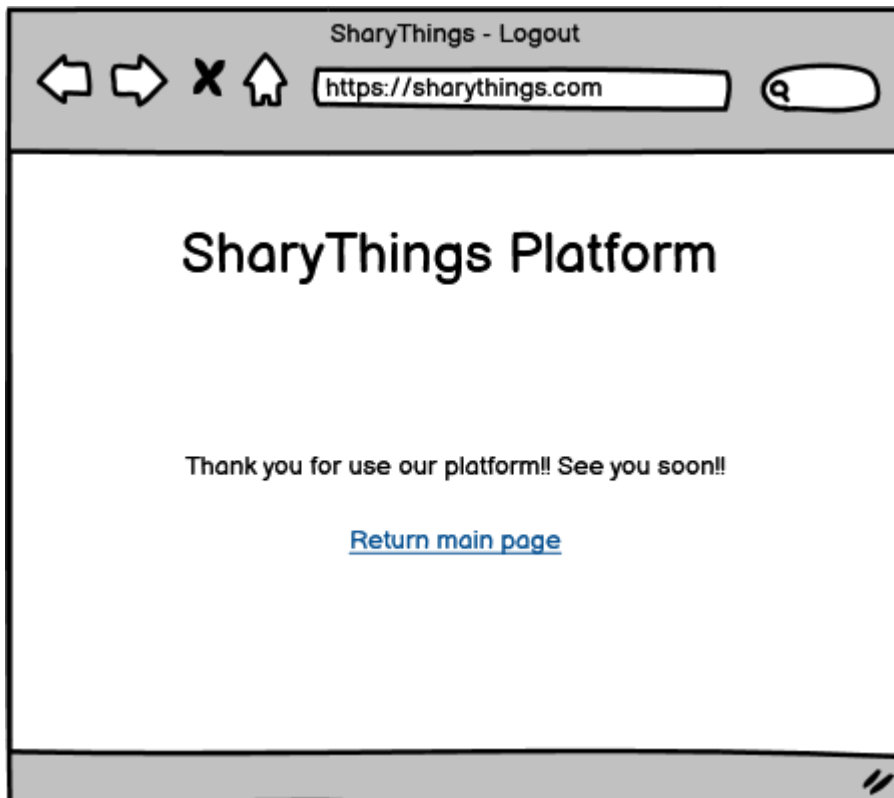
https://sharythings.com

SharyThings Platform

User/mail

Password

UserLogout



A hand-drawn sketch of a web browser window titled "SharyThings - Logout". The address bar shows "https://sharythings.com". The main content area displays "SharyThings Platform" in a large font. Below this, there is a message: "Thank you for use our platform!! See you soon!!". A blue link labeled "Return main page" is centered below the message. The browser window has a grey header and footer bar.

SharyThings - Logout

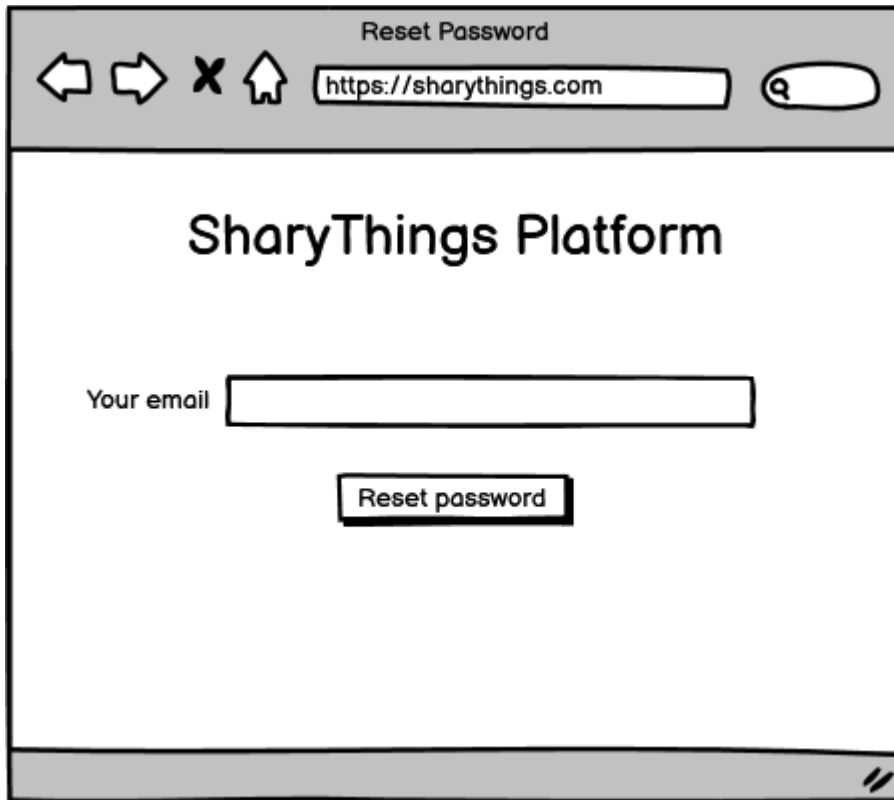
https://sharythings.com

SharyThings Platform

Thank you for use our platform!! See you soon!!

[Return main page](#)

RecoveryPassword



Reset Password

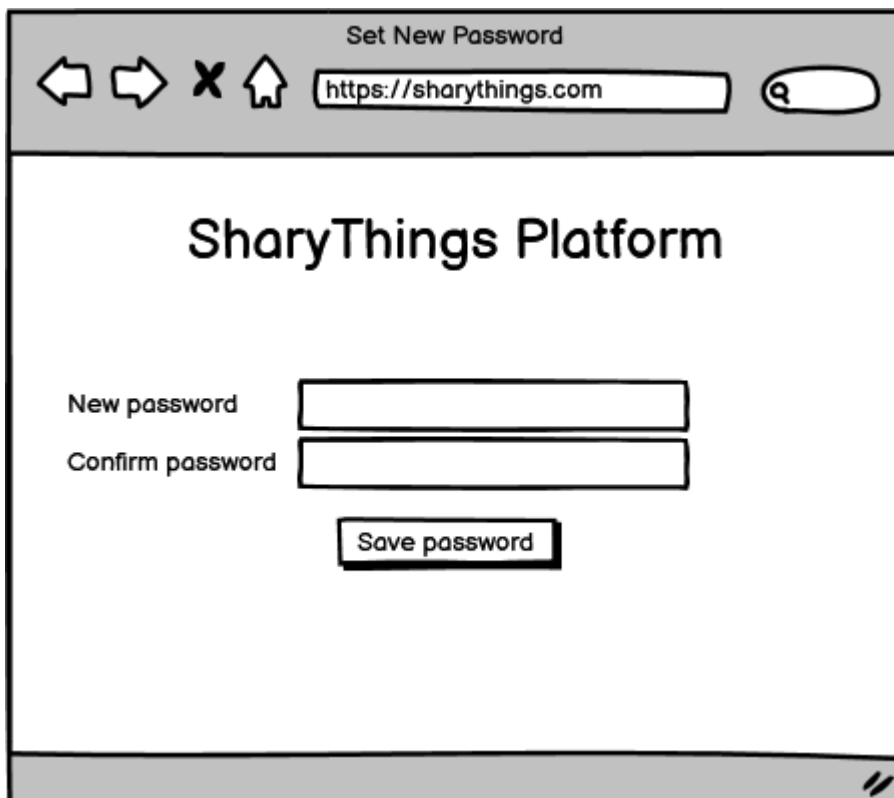
https://sharythings.com

SharyThings Platform

Your email

This is a hand-drawn diagram of a web browser window titled "Reset Password". The address bar shows "https://sharythings.com". The main content area displays the "SharyThings Platform" heading, followed by a label "Your email" and a text input field. Below the input field is a button labeled "Reset password". The browser window includes standard navigation icons (back, forward, stop, home) and a search icon.

SetNewPassword



Set New Password

https://sharythings.com

SharyThings Platform

New password

Confirm password

This is a hand-drawn diagram of a web browser window titled "Set New Password". The address bar shows "https://sharythings.com". The main content area displays the "SharyThings Platform" heading, followed by two labels: "New password" and "Confirm password", each with a corresponding text input field. Below the input fields is a button labeled "Save password". The browser window includes standard navigation icons (back, forward, stop, home) and a search icon.

ShowUserProfile

SharyThings - User Profile
https://sharythings.com

SharyThings Platform

Name Marty **Surname** McFly **Mall** backtothe@future.com
Address 9303 Lyon Drive, Lyon Estates, Hill Valley CA 95420
Birth Date 01/01/1972 **Phone** +1 555-0199

Products

Name^	Family	SubFamily	Status^v	View
product1	Family1	SubFamily1	ToShary	<input checked="" type="checkbox"/>
product2	Family2	SubFamily2	UnShary	<input checked="" type="checkbox"/>
product3	Family3	SubFamily3	Borrowed	<input checked="" type="checkbox"/>

Borrows

ProductName^	Lender^	Family	SubFamily	Status	View
product1	Lender1	Family1	SubFamily1	InProgres	<input checked="" type="checkbox"/>
product2	Lender2	Family2	SubFamily2	Finished	<input checked="" type="checkbox"/>
product3	Lender3	Family3	SubFamily3	PendingApproval	<input checked="" type="checkbox"/>
product4	Lender4	Family4	SubFamily4	Rejected	<input checked="" type="checkbox"/>

OK

CreateProduct

SharyThings - Create product

https://sharythings.com

SharyThings Platform


Name

Family

Subfamily

Availability

Observations



ShowProductDetail

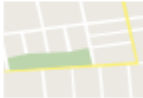
Show Product Detail

https://sharythings.com

SharyThings Platform

Name

Description

Photos Position 

Availiability

UnsharedProduct

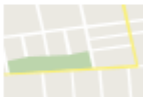
Show Product Detail

https://sharythings.com

SharyThings Platform

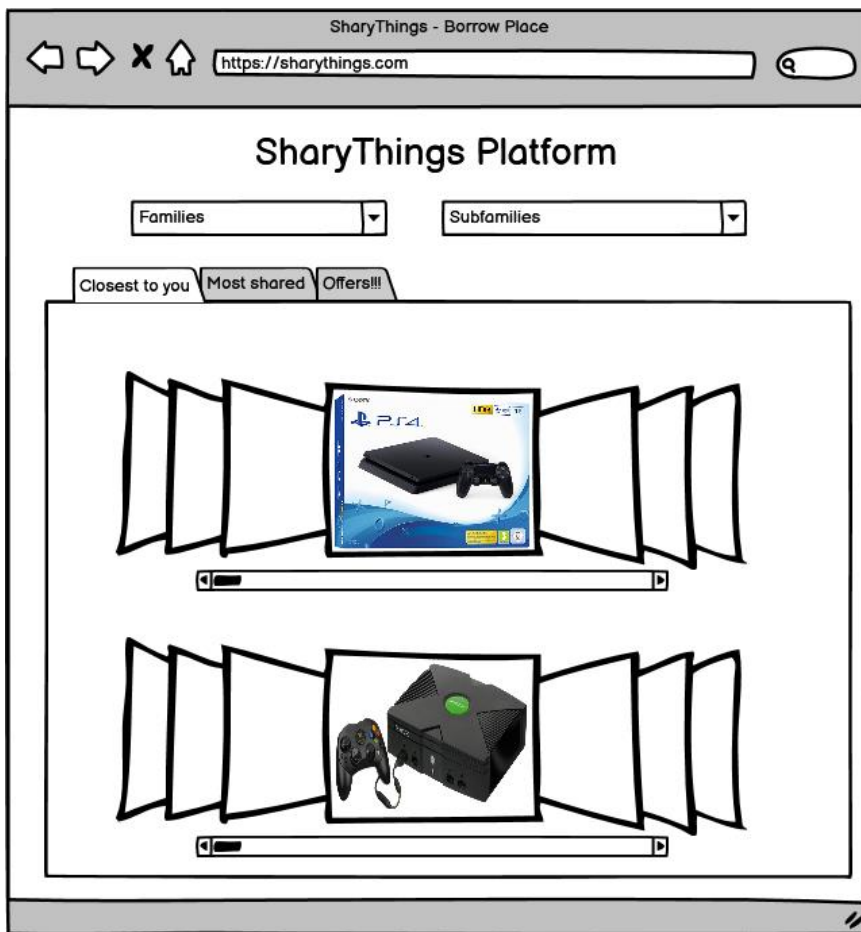
Name

Description

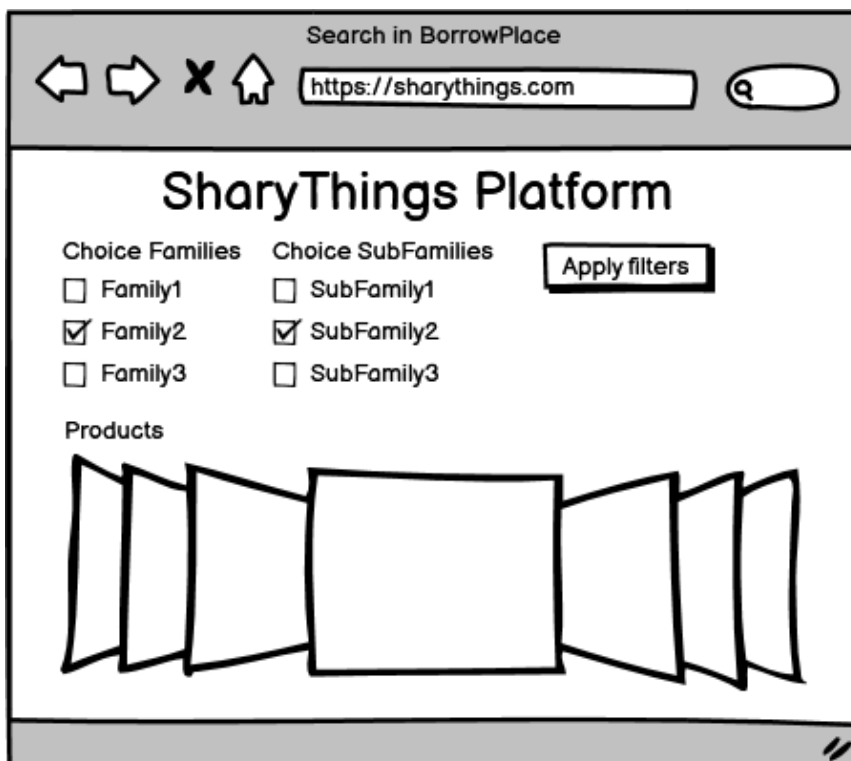
Photos Position 

Availiability

AccessBorrowPlace



SearchInBorrowPlace



CreateBorrow

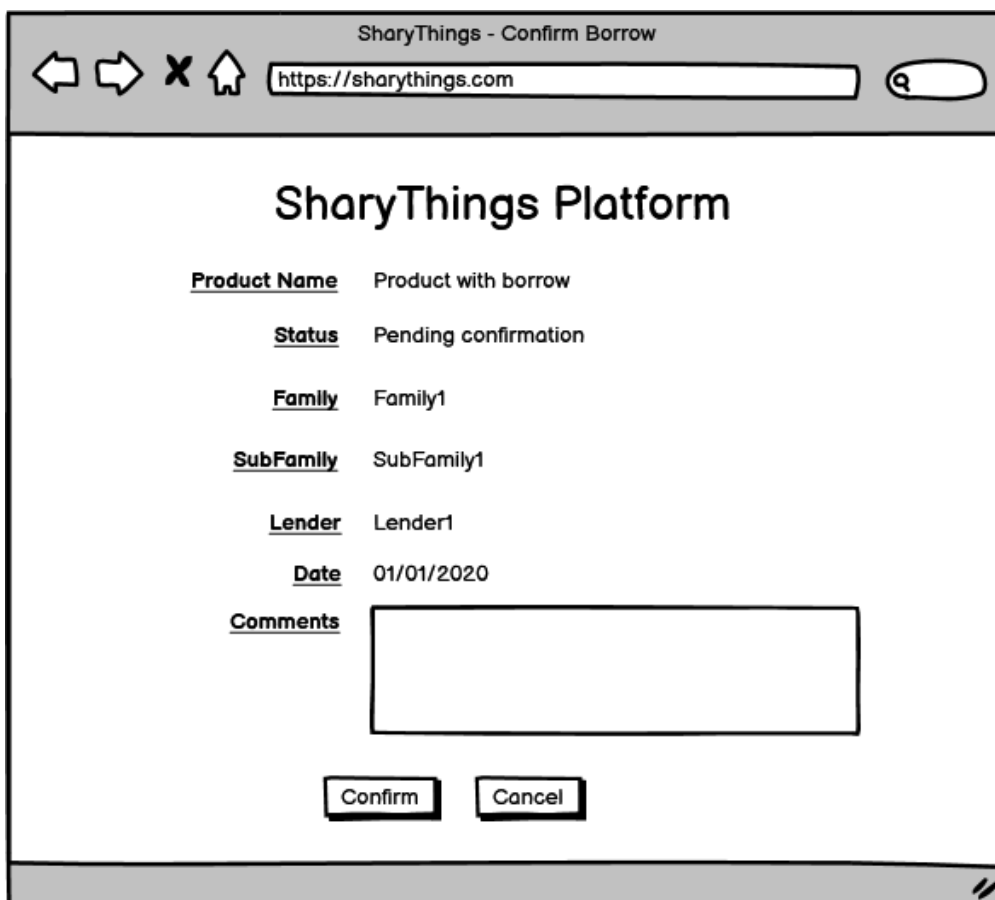


A browser window titled "SharyThings - Borrow Request" showing a form for creating a borrow request. The browser address bar displays "https://sharythings.com". The form has a title "SharyThings Platform" and contains the following fields:

- Product Name**: Product with borrow
- Lender**: Lender1
- Date**: 01/01/2020
- Product Location**: A map showing a street grid with a green highlighted area and a yellow line.

At the bottom of the form are two buttons: "Request" and "Cancel".

ConfirmBorrow



A browser window titled "SharyThings - Confirm Borrow" showing a form for confirming a borrow request. The browser address bar displays "https://sharythings.com". The form has a title "SharyThings Platform" and contains the following fields:

- Product Name**: Product with borrow
- Status**: Pending confirmation
- Family**: Family1
- SubFamily**: SubFamily1
- Lender**: Lender1
- Date**: 01/01/2020
- Comments**: A large empty text box.

At the bottom of the form are two buttons: "Confirm" and "Cancel".

AddCommentToBorrow

A browser window titled "Add comment to borrow" with a URL bar showing "https://sharythings.com". The page content includes the "SharyThings Platform" header, a label "Borrow xxxxxxxxxx", a "Comment" label next to a large text input area, and two buttons at the bottom: "Save comment" and "Close".

RejectBorrow

A browser window titled "SharyThings - Borrow Reject" with a URL bar showing "https://sharythings.com". The page content includes the "SharyThings Platform" header and a table with the following data:

<u>Product Name</u>	Product with borrow
<u>Status</u>	Pending approval
<u>Borrower</u>	Borrower1
<u>Date</u>	01/01/2020

At the bottom of the page are three buttons: "Approval", "Reject", and "Cancel".

SetScoreBorrow

SharyThings - Set score borrow

https://sharythings.com

SharyThings Platform

Product Name Product with borrow

Status Finished

Family Family1

Lender Lender1

Score

ShowBorrowList

Borrows User

https://sharythings.com

SharyThings Platform

User

Borrows

ProductName^	Lender^	Family	SubFamily	Status	View
product1	Lender1	Family1	SubFamily1	InProgres	<input checked="" type="checkbox"/>
product2	Lender2	Family2	SubFamily2	Finished	<input checked="" type="checkbox"/>
product3	Lender3	Family3	SubFamily3	PendingApproval	<input checked="" type="checkbox"/>
product4	Lender4	Family4	SubFamily4	Rejected	<input checked="" type="checkbox"/>

ShowBorrowDetail

The screenshot shows a web browser window titled "SharyThings - Borrow Detail" with the URL "https://sharythings.com". The page header is "SharyThings Platform". The main content area displays the following details for a borrow:

- Product Name:** Product with borrow
- Status:** In progress
- Family:** Family1
- SubFamily:** SubFamily1
- Lender:** Lender1
- Date:** 12/12/2021 (with a calendar icon)
- Comments:** A large empty text box for comments.

At the bottom of the form is an "OK" button.

ShowUserScore

The screenshot shows a web browser window titled "SharyThings - Show user score" with the URL "https://sharythings.com". The page header is "SharyThings Platform". The main content area displays the following details for a user's score:

- Product Name:** Product with borrow
- Status:** Finished
- Family:** Family1
- Lender:** Lender1
- Score:** A horizontal bar chart showing a score of approximately 75% (the bar is 75% green and 25% white).

At the bottom of the form is a "Close" button.

ShowProductList


The screenshot shows a web browser window titled "Products User" with the address bar displaying "https://sharythings.com". The main content area is titled "SharyThings Platform" and contains a "User" input field. Below this is a section titled "Products" which contains a table with the following data:

Name^	Family	SubFamily	Status^v	View
product1	Family1	SubFamily1	ToShary	<input checked="" type="checkbox"/>
product2	Family2	SubFamily2	UnShary	<input checked="" type="checkbox"/>
product3	Family3	SubFamily3	Borrowed	<input checked="" type="checkbox"/>

Below the table is a "Close" button.

UpdateUserProfile

The screenshot shows a web browser window titled "SharyThings - Update User Data" with the address bar displaying "https://sharythings.com". The main content area is titled "SharyThings Platform" and contains a form with the following fields:

- Name:
- Surname:
- Address:
- Birth Date: / / 
- Phone:
- Password:

At the bottom of the form are two buttons: "Update" and "Cancel".

EditBorrow

SharyThings - Borrow Detail

https://sharythings.com

SharyThings Platform



Product Name Product with borrow

Status In progress

Family Family1

SubFamily SubFamily1

Lender Lender1

Date 12/12/2021  12/12/2021 

Comments

OK

DeleteProduct

SharyThings - Delete Product

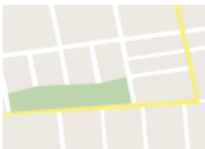
https://sharythings.com

SharyThings Platform

Name product name

Description Description product, Description product, Description product

Photos

Position 

Availability 10/10/2021 31/10/2021

Delete Close

EditProduct

SharyThings - Edit Product

https://sharythings.com

Q

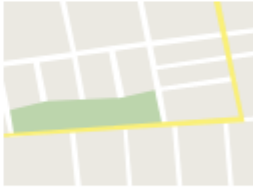
SharyThings Platform

Name

Description

Photos

Position



Availability

//

//

let's make a borrow

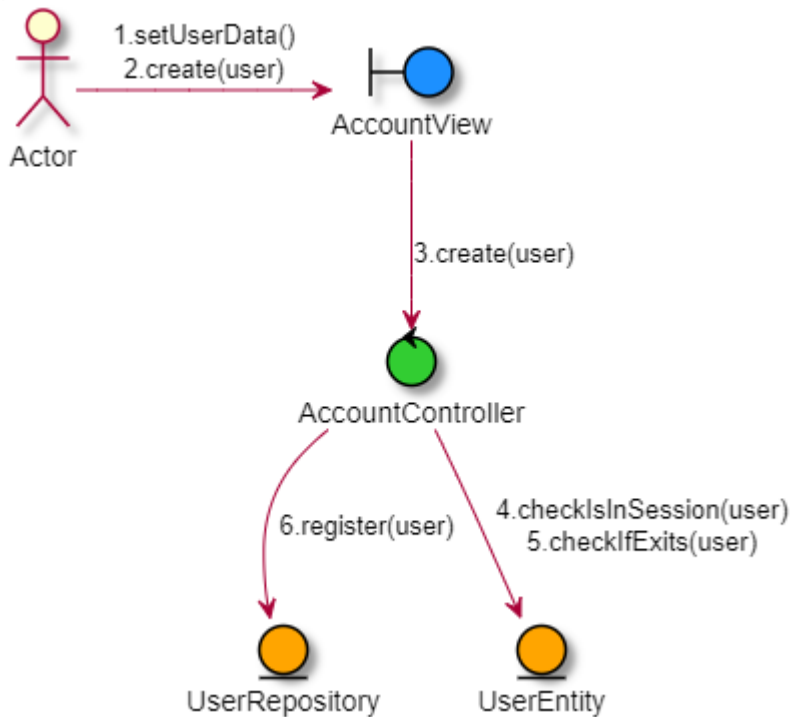
unshare / share

Close

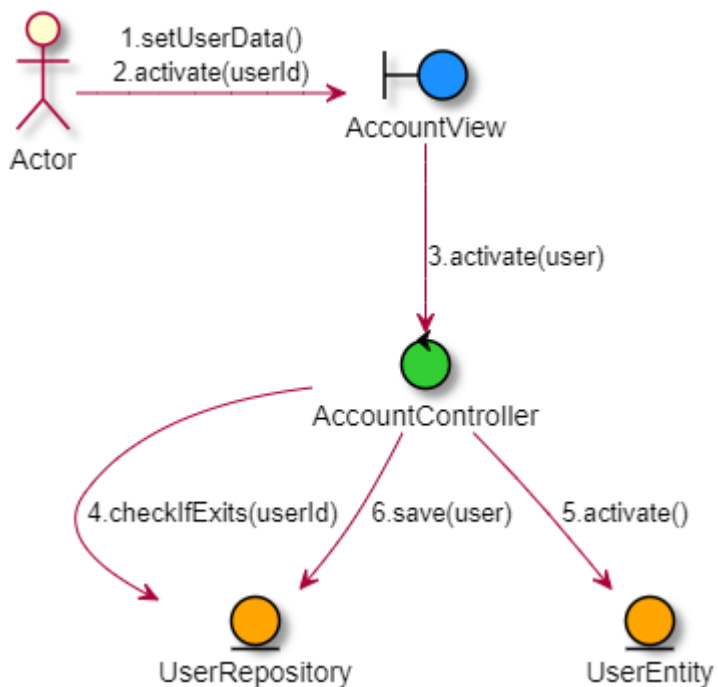
Análisis

El propósito del análisis es examinar detalladamente a través de la captura de requisitos mediante su refinamiento y estructuración para lograr una comprensión más precisa de los requisitos.

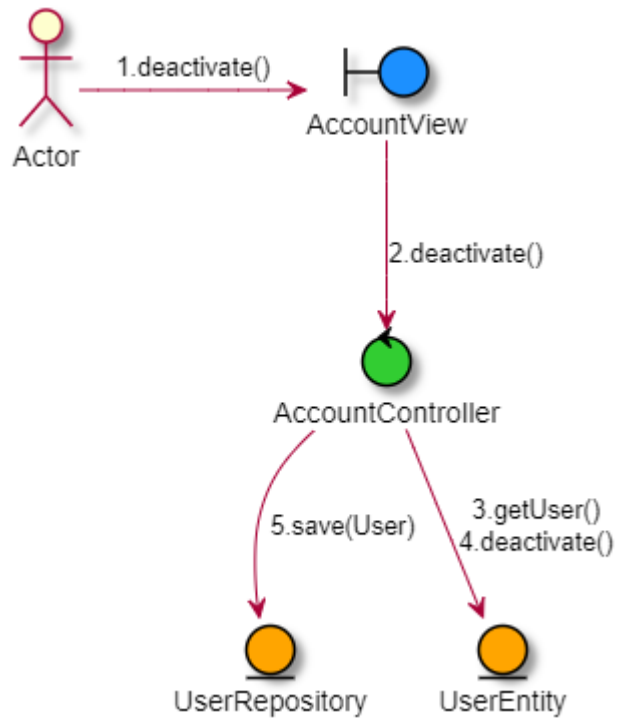
CreateUser



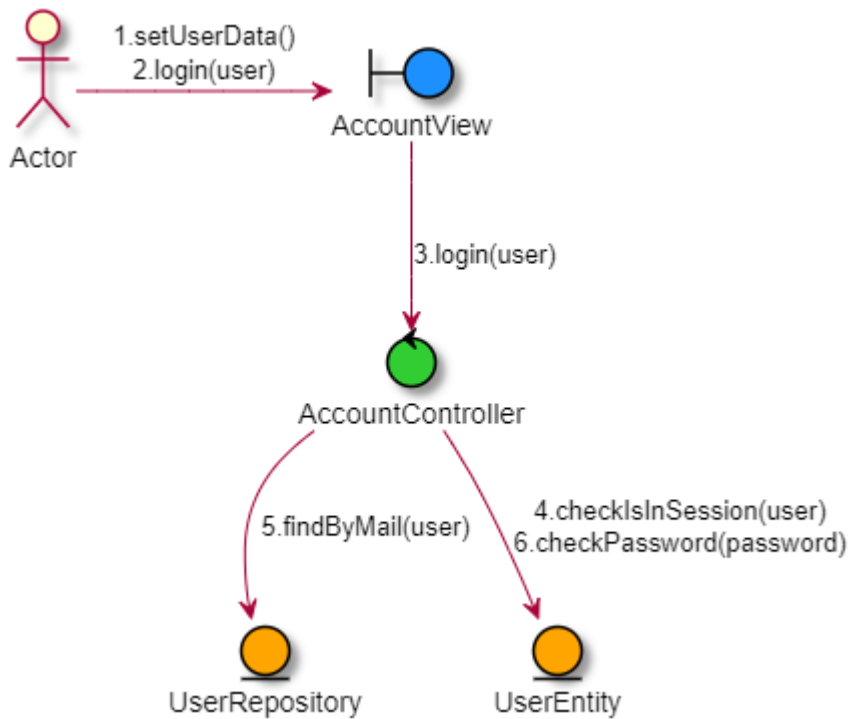
ActivateUser



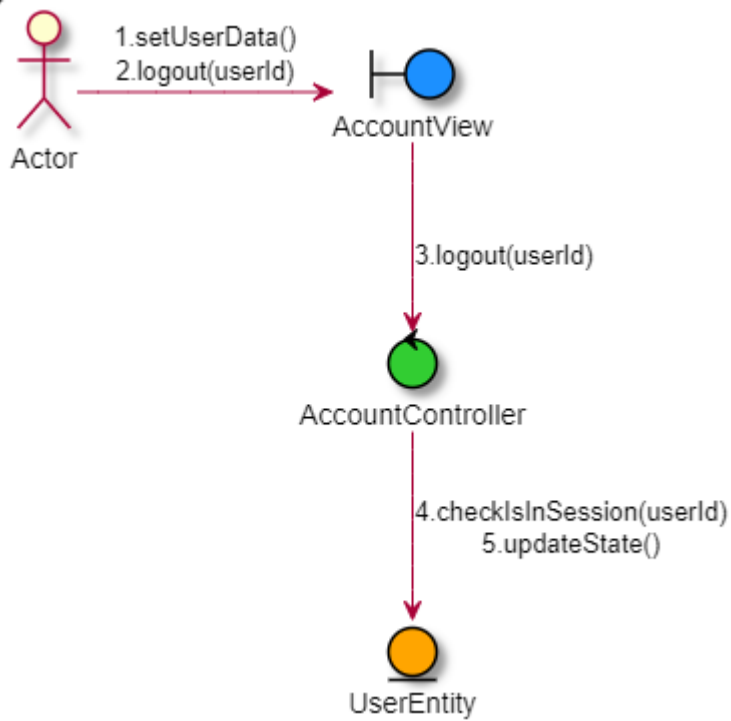
DeactivateUser



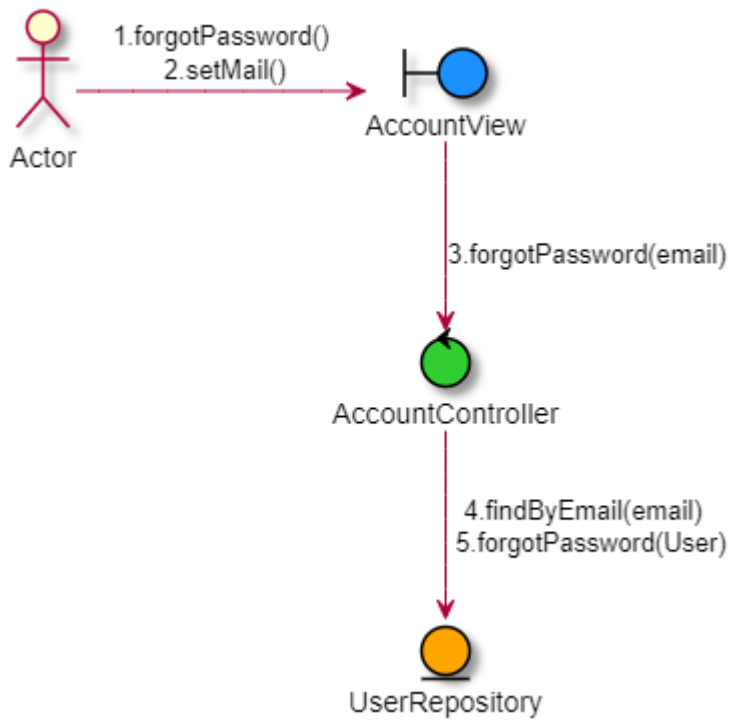
UserLogin



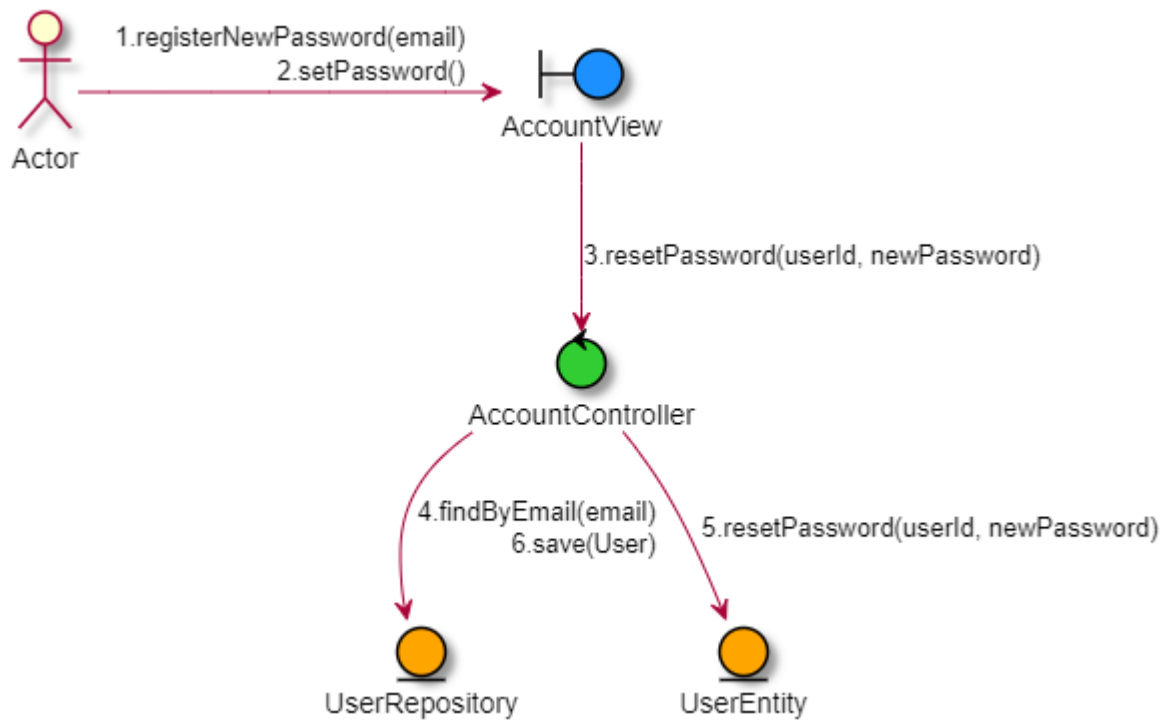
UserLogout



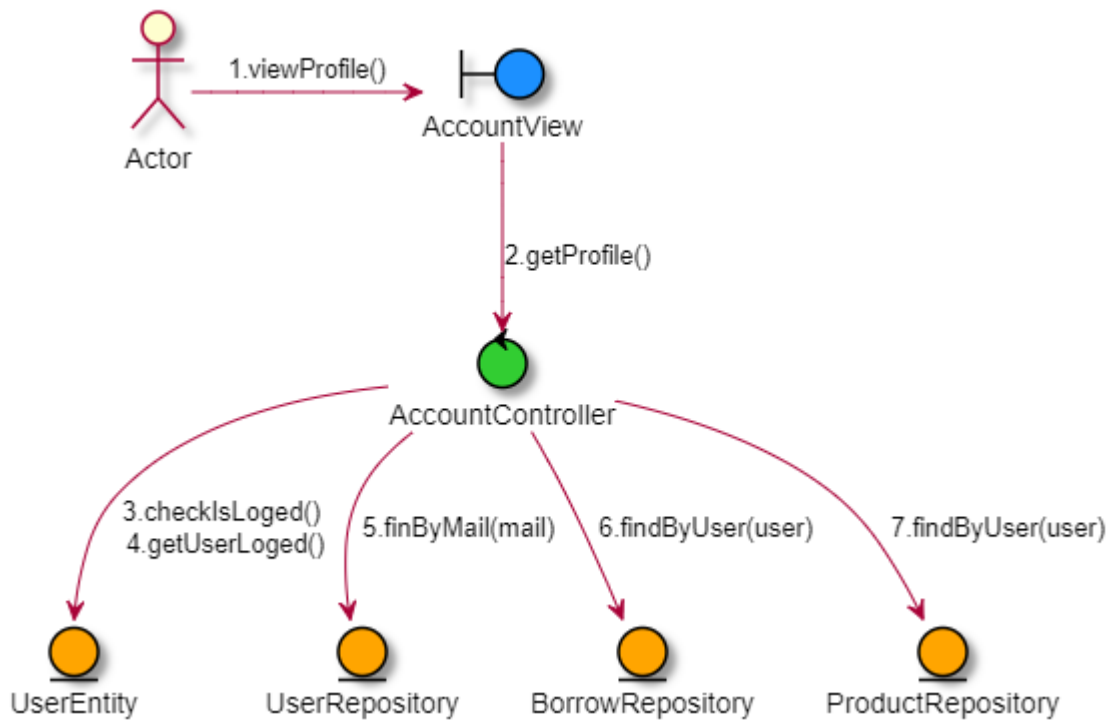
RecoveryPassword



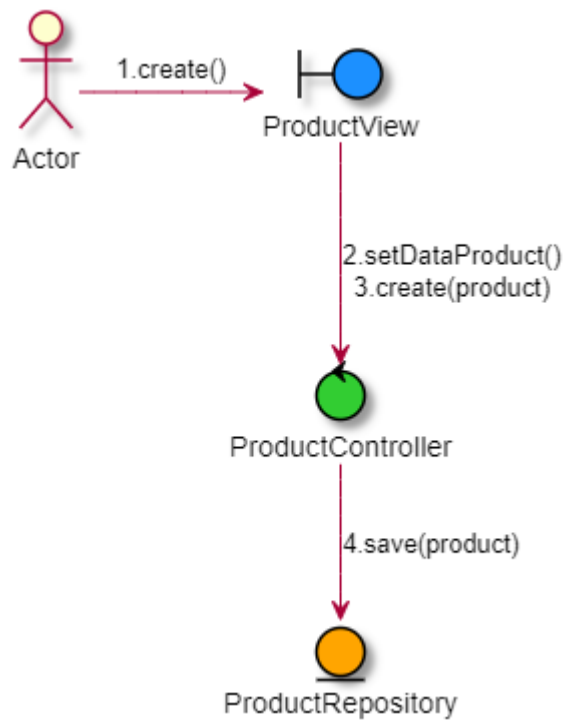
SetNewPassword



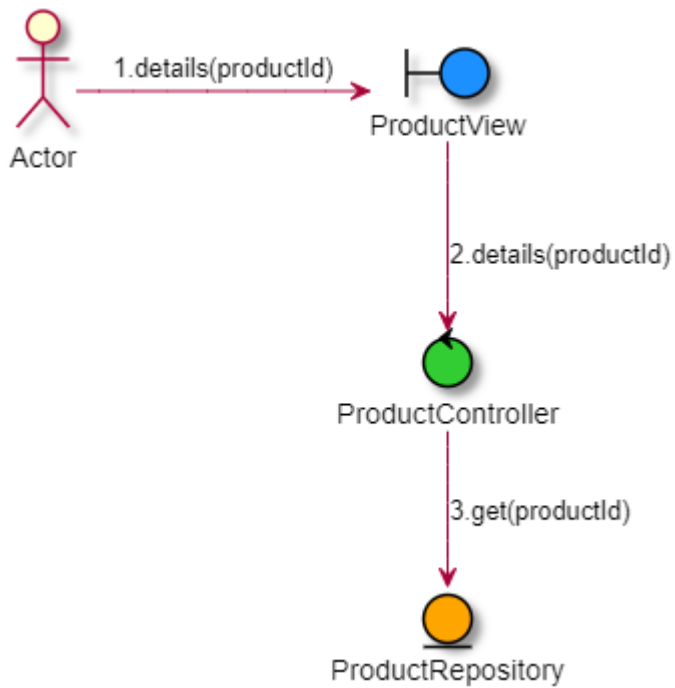
ShowUserProfile



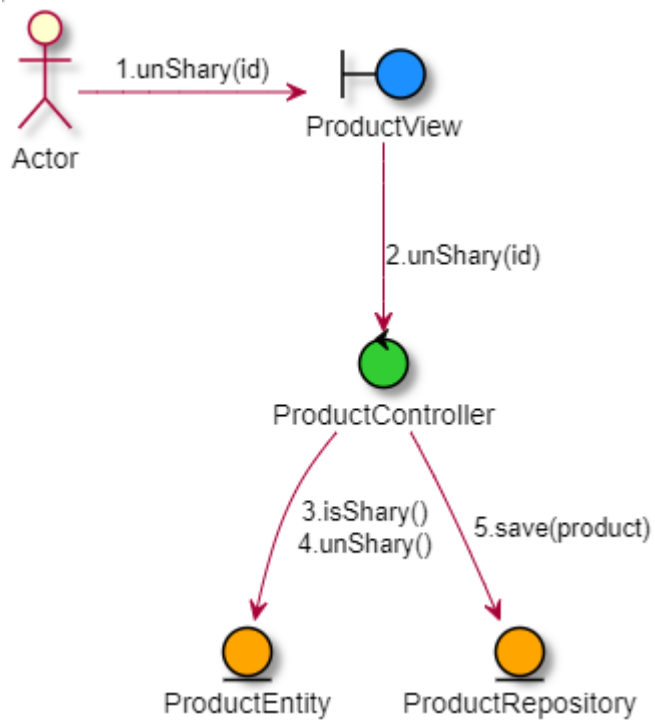
CreateProduct



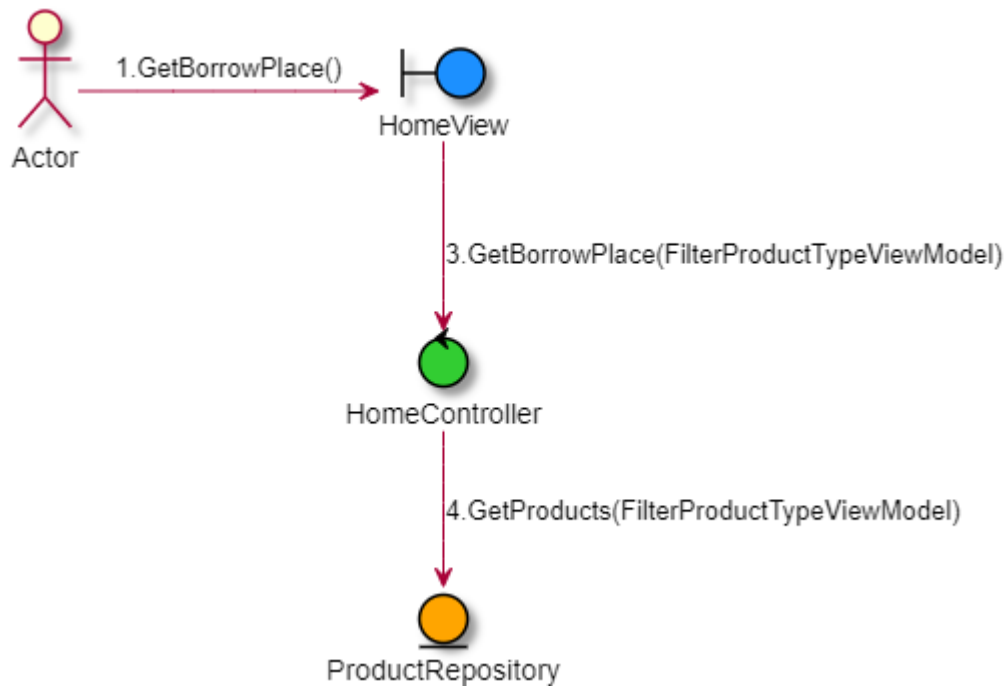
ShowProductDetail



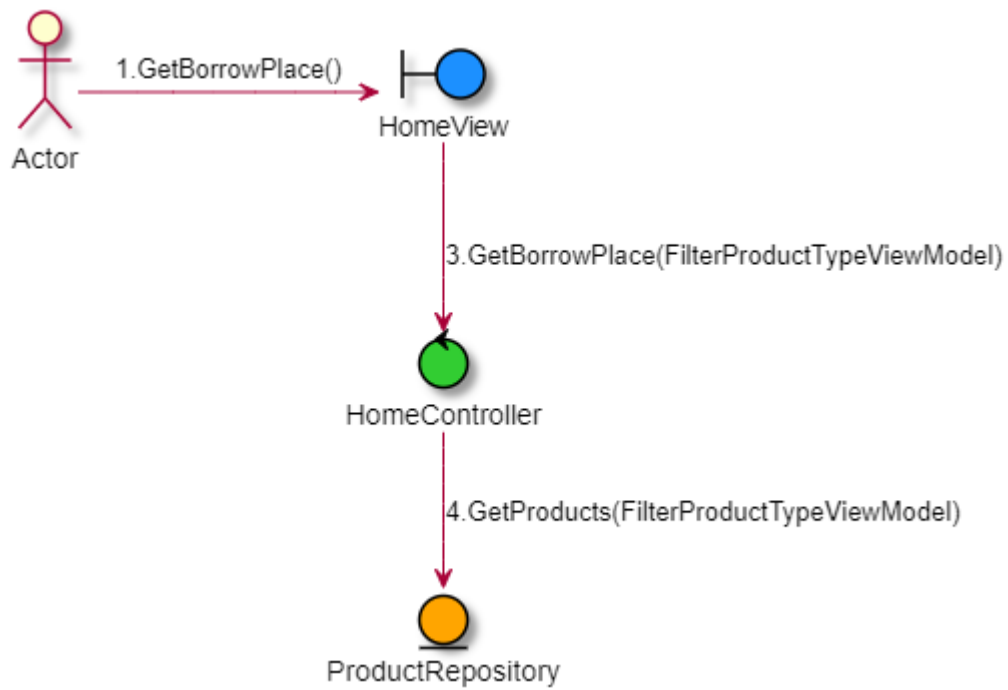
UnsharedProduct



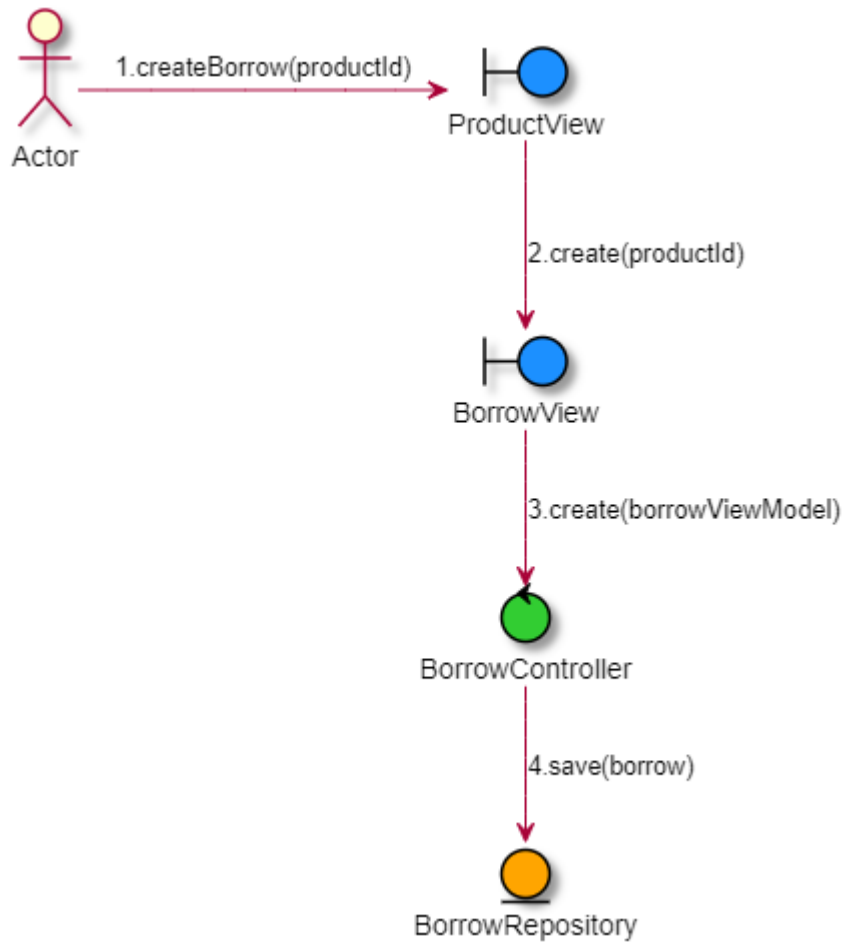
AccessBorrowPlace



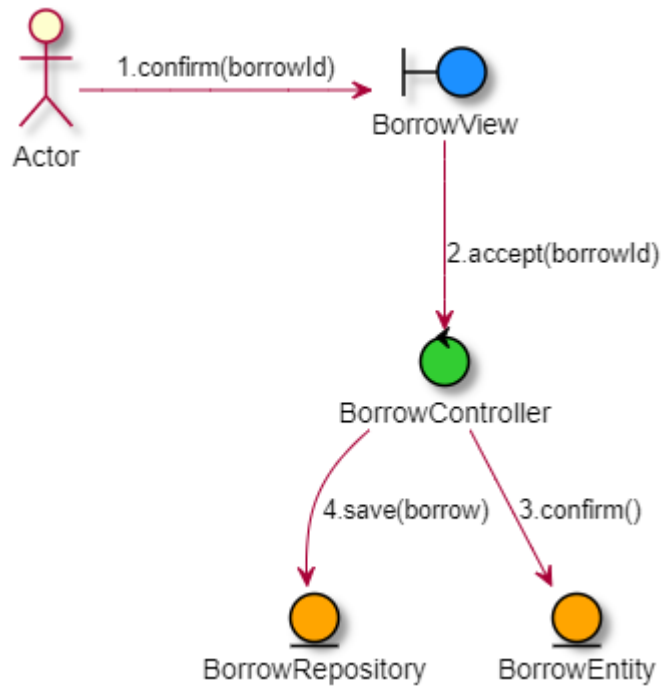
SearchInBorrowPlace



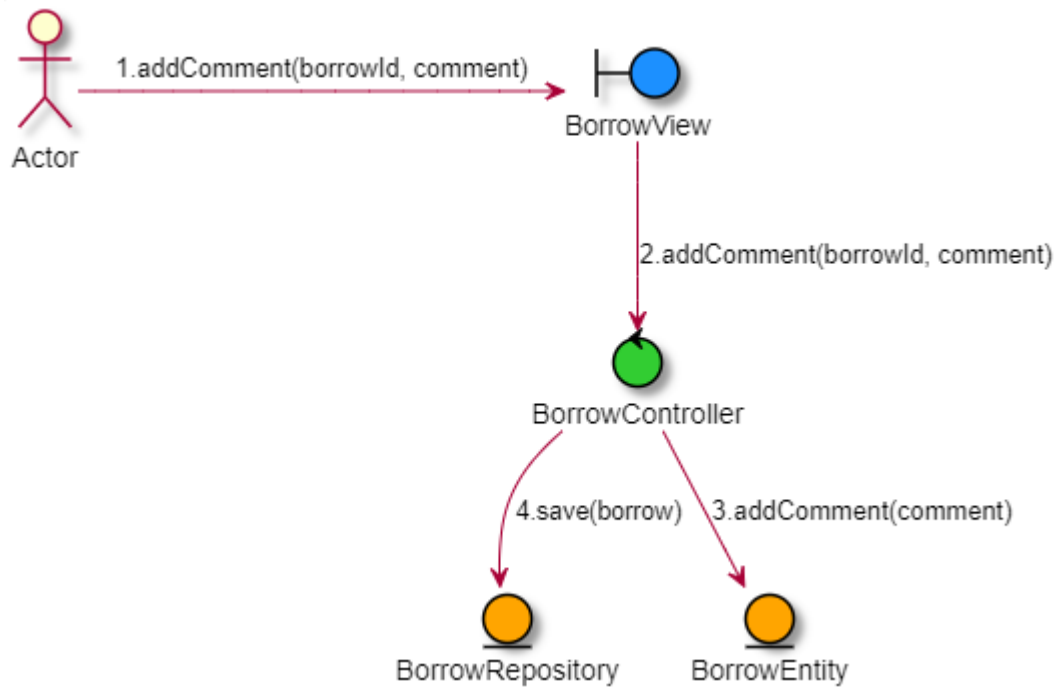
CreateBorrow



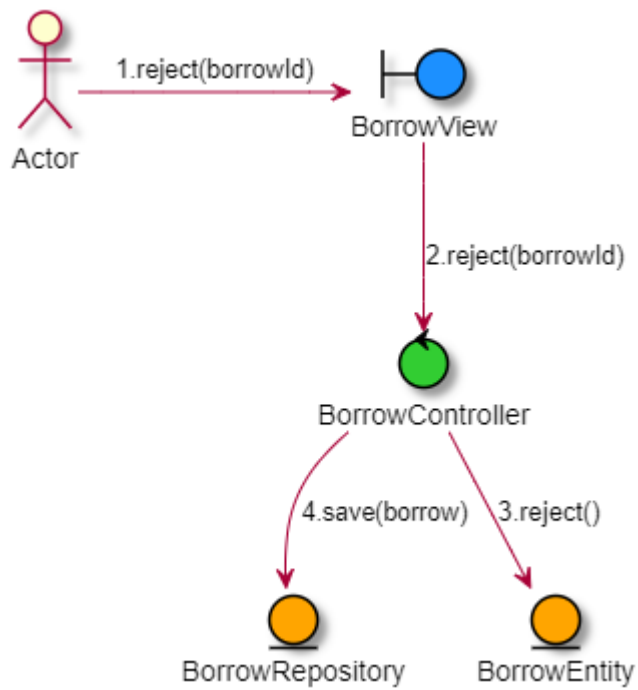
ConfirmBorrow



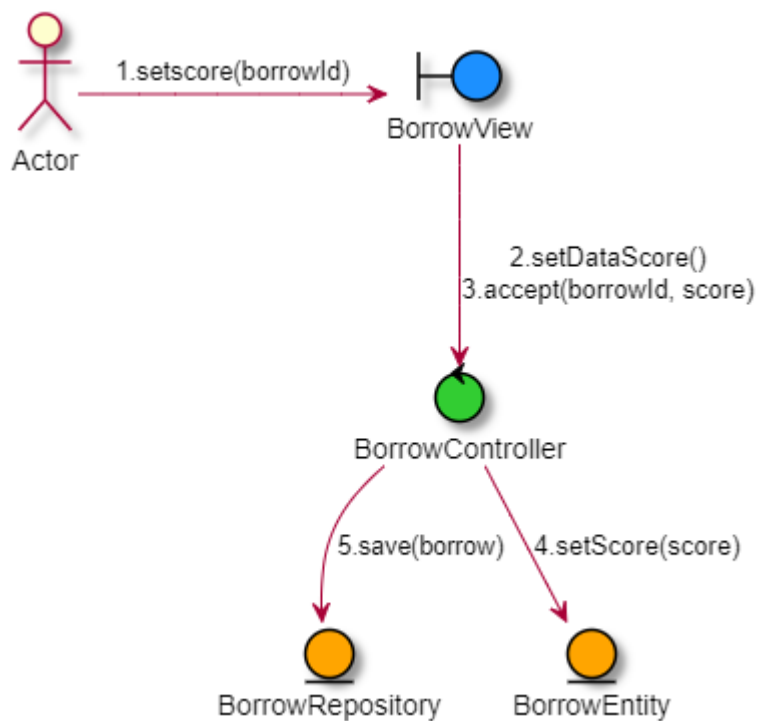
AddCommentToBorrow



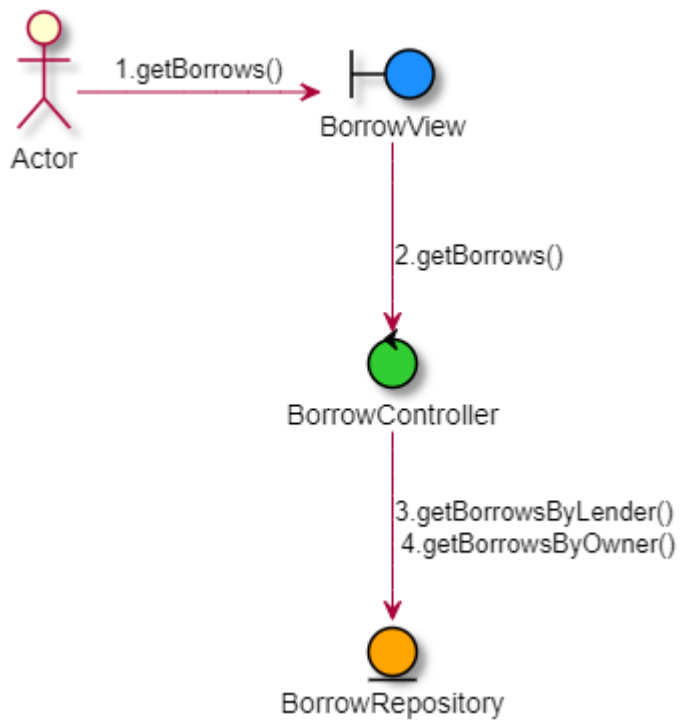
RejectBorrow



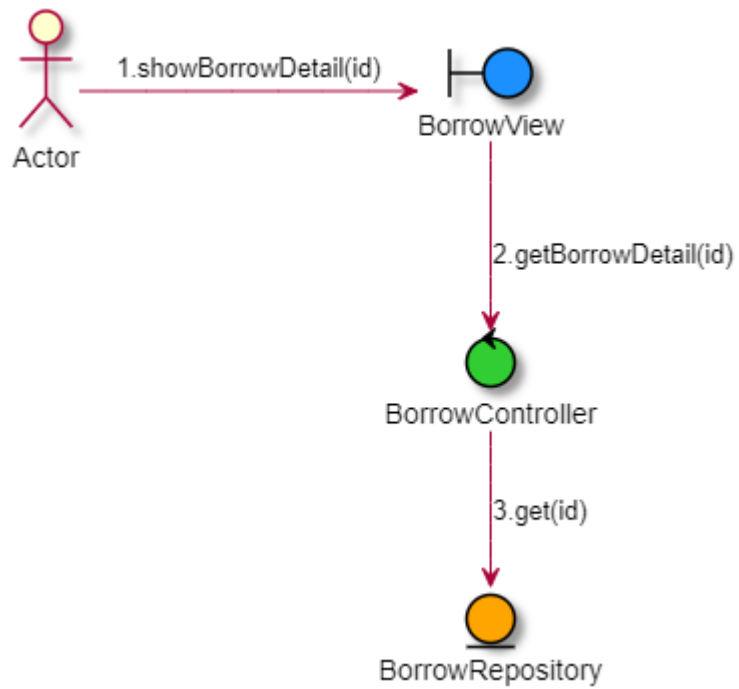
SetScoreBorrow



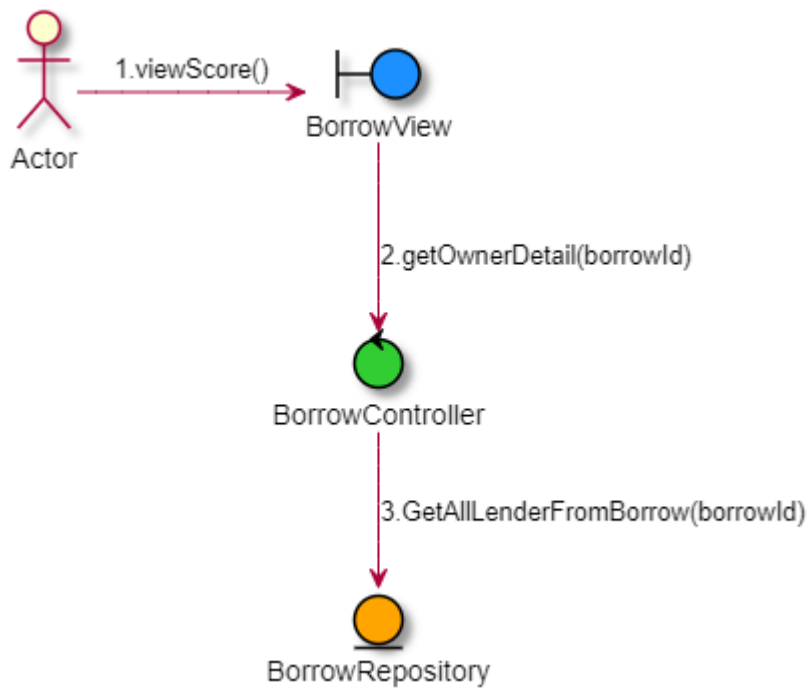
ShowBorrowList



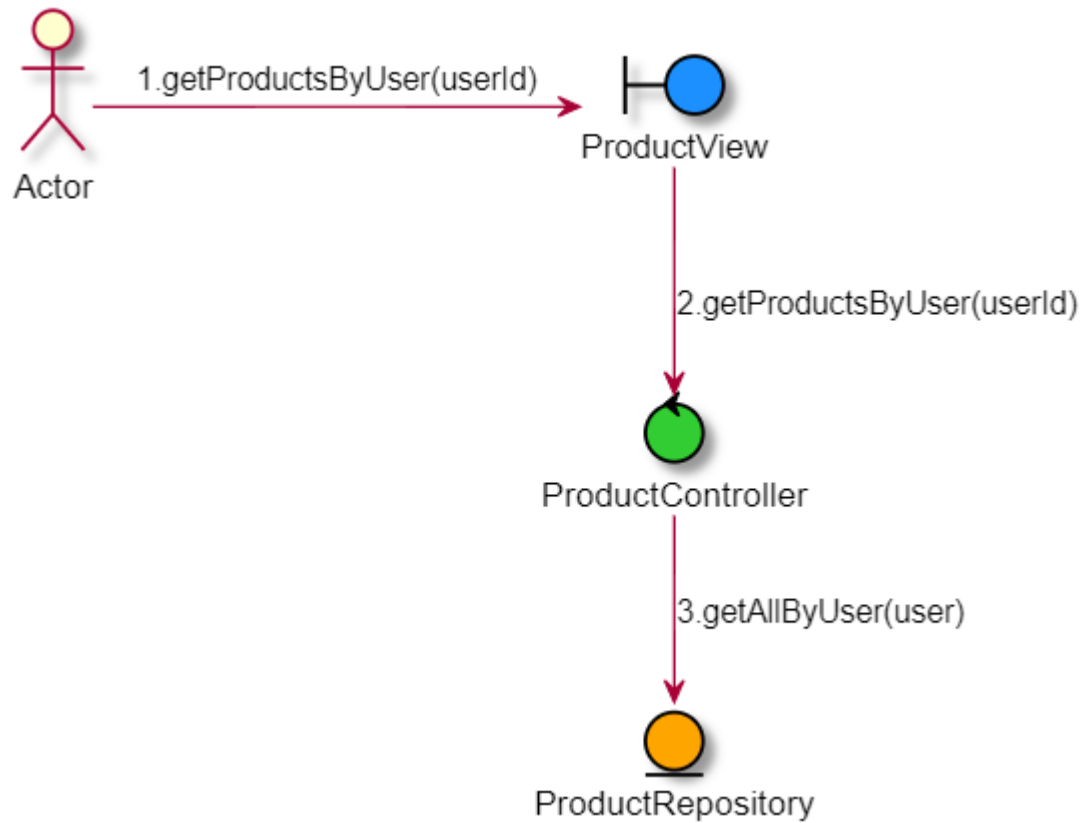
ShowBorrowDetail



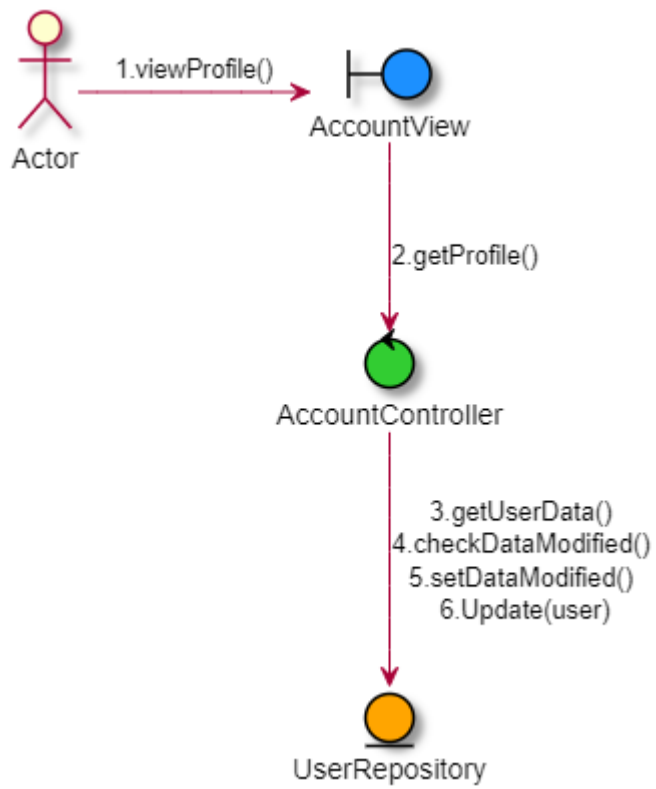
ShowUserScore



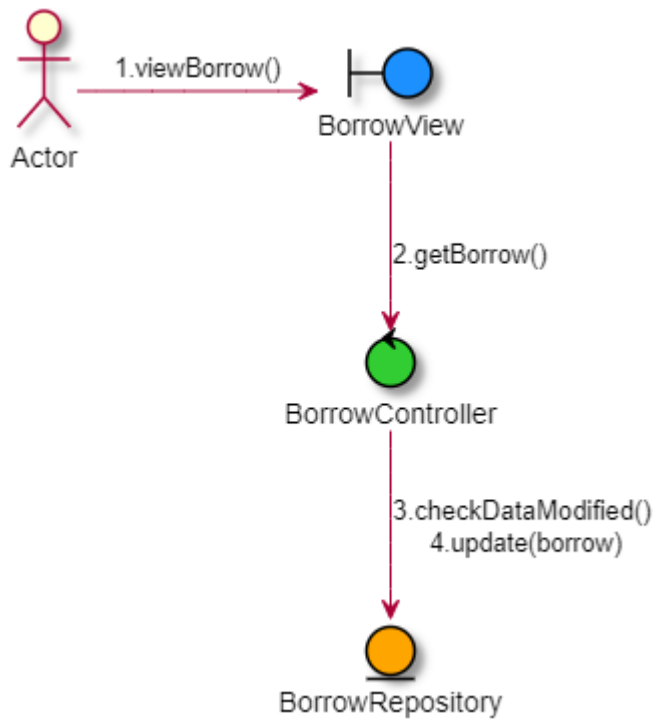
ShowProductList



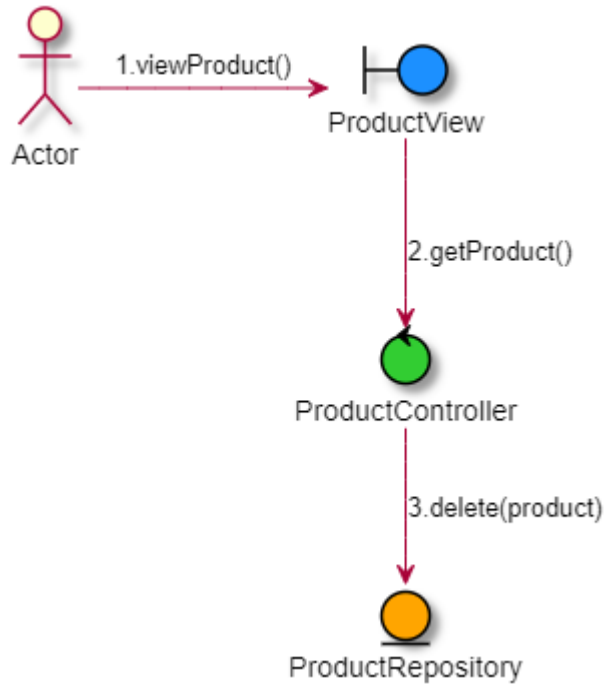
UpdateUserProfile



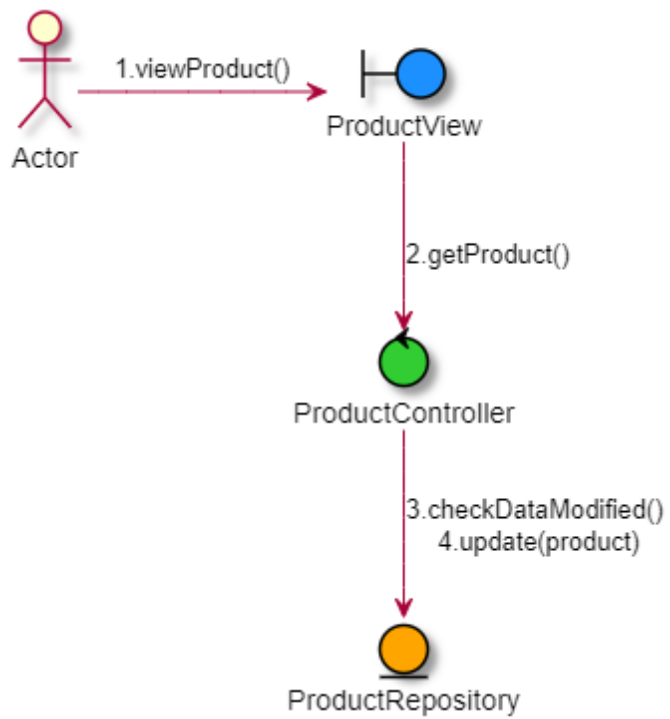
EditBorrow



DeleteProduct



EditProduct



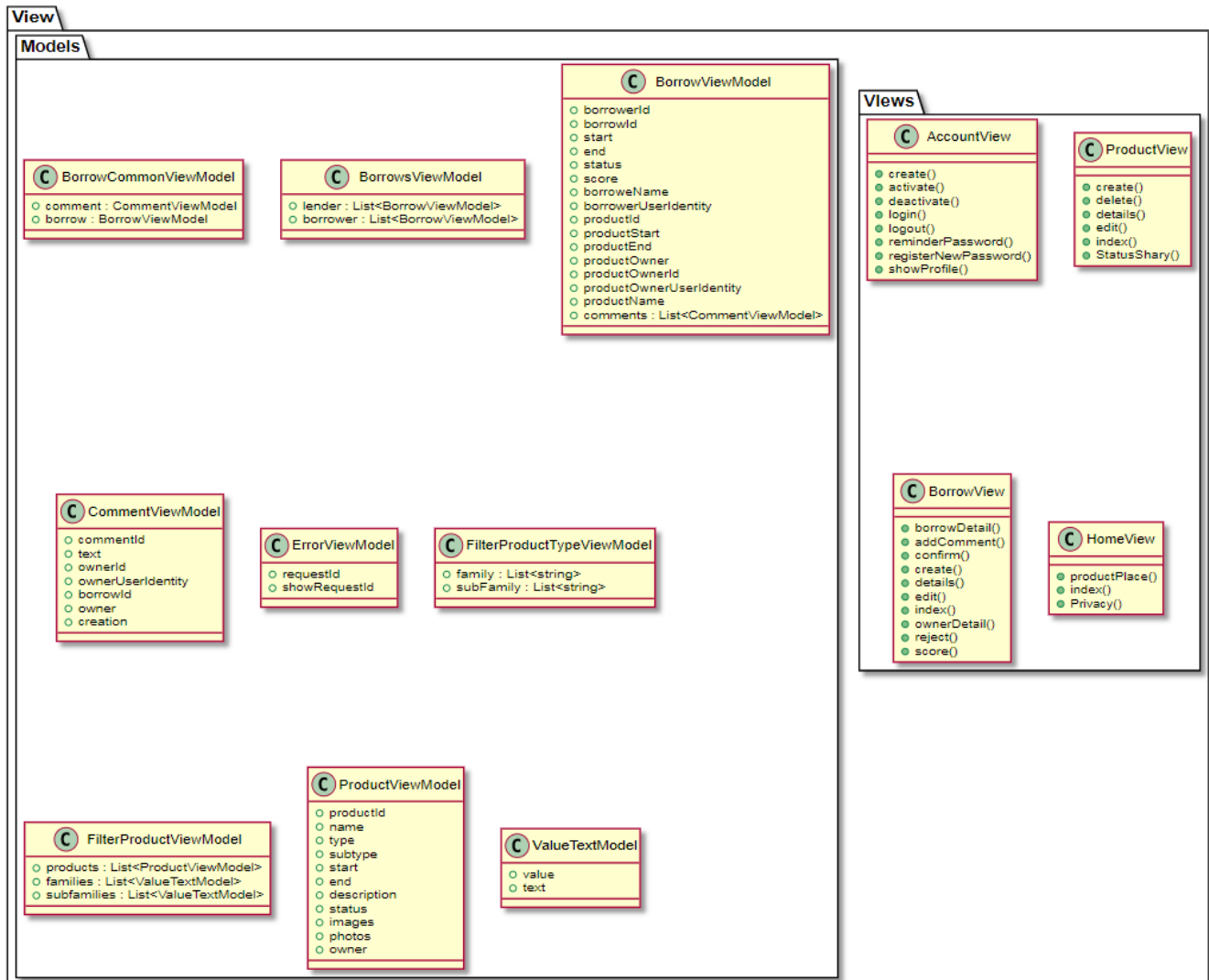
Diseño

El propósito del diseño es desarrollar modelos enfocados sobre los requisitos no funcionales y el dominio de la solución y preparar para la implementación y pruebas del sistema.

A continuación, se verán tres (3) apartados, diseño de la arquitectura, diseño de la capa de datos y diseño de los casos de uso.

Diseño de la arquitectura

En este apartado se presentan el análisis de la arquitectura del sistema, la cual establece la organización del sistema software y los elementos estructurales en los que el sistema está compuesto.



Controllers

C HomeController

- GetBorrowPlace(FilterProductTypeViewModel)
- Privacy()

C ProductsController

- GetProducts() : List<ProductViewModel>
- Details(idProduct) : ProductViewModel
- Create(ProductViewModel)
- Edit(idProduct, ProductViewModel)
- Delete(idProduct)
- DeletePhoto(idProduct, idPhoto)
- ChangeStatusShary(idProduct)

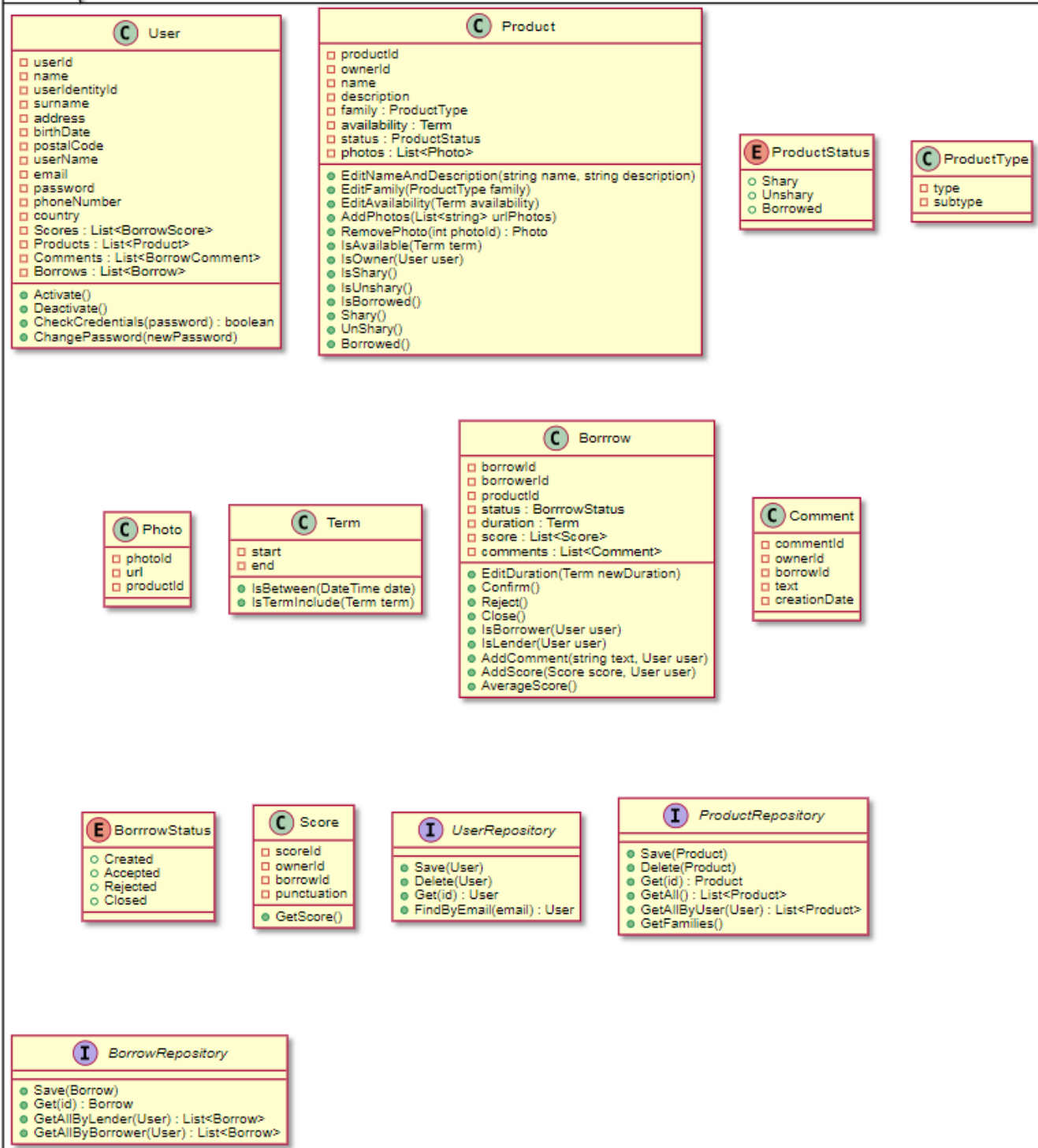
C BorrowsController

- GetBorrows() : List<BorrowsViewModel>
- Details(idBorrow) : BorrowsViewModel
- Create(idProduct, BorrowsViewModel)
- Edit(idBorrow, BorrowsViewModel)
- AddComment(idBorrow, BorrowsViewModel)
- Confirm(idBorrow)
- Reject(idBorrow)
- Score(idBorrow, BorrowsViewModel)
- OwnerDetail(idBorrow) : List<BorrowsViewModel>

C AccountController

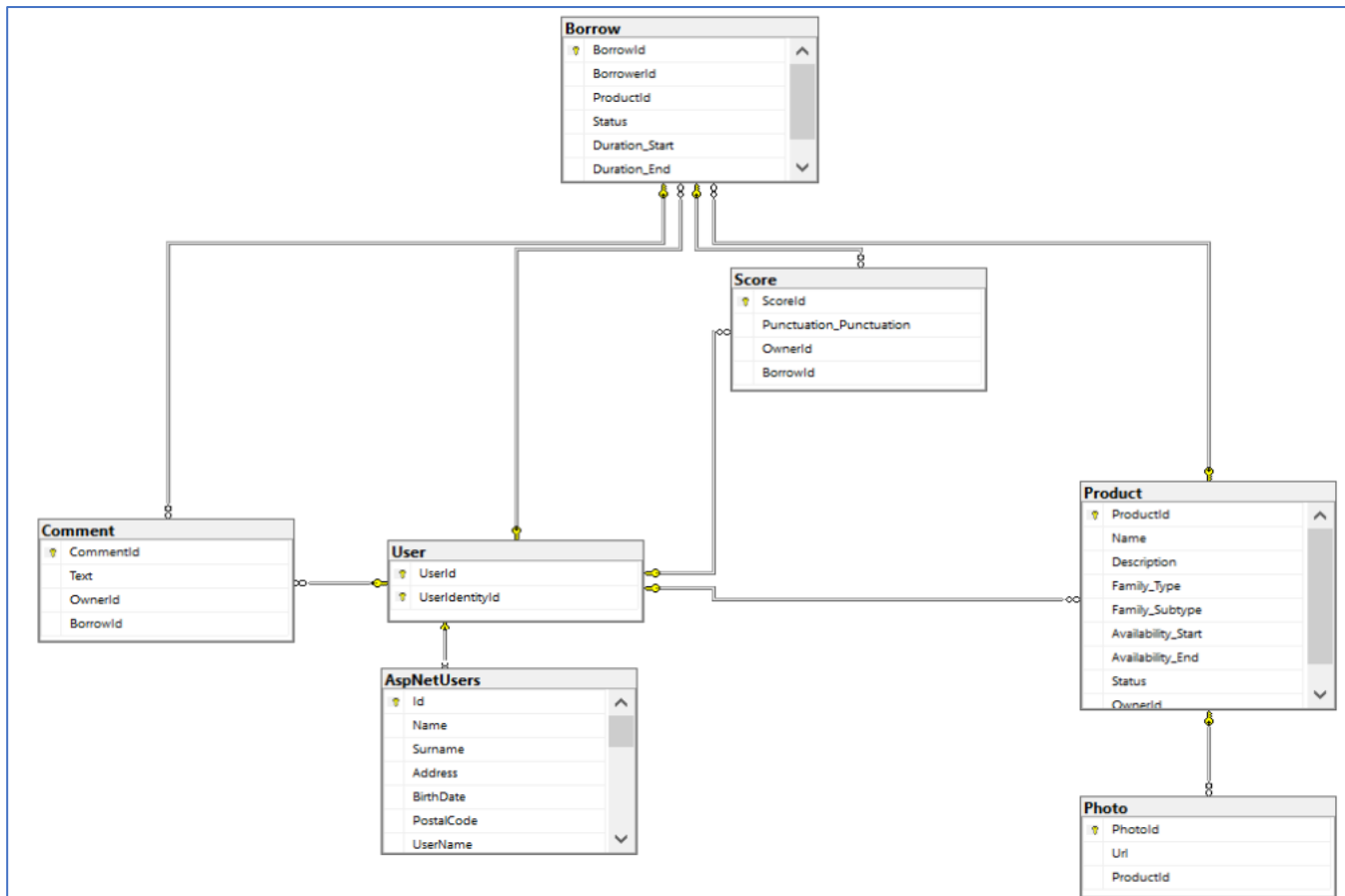
- Register(UserViewModel)
- Activate(userId)
- Deactivate(userId)
- Login(email, password)
- Logout(userId)
- ForgotPassword(email)
- ResetPassword(userId, newPassword)
- ResendEmailConfirmation(email)
- ChangeEmail(email)
- GetProfile(userId)

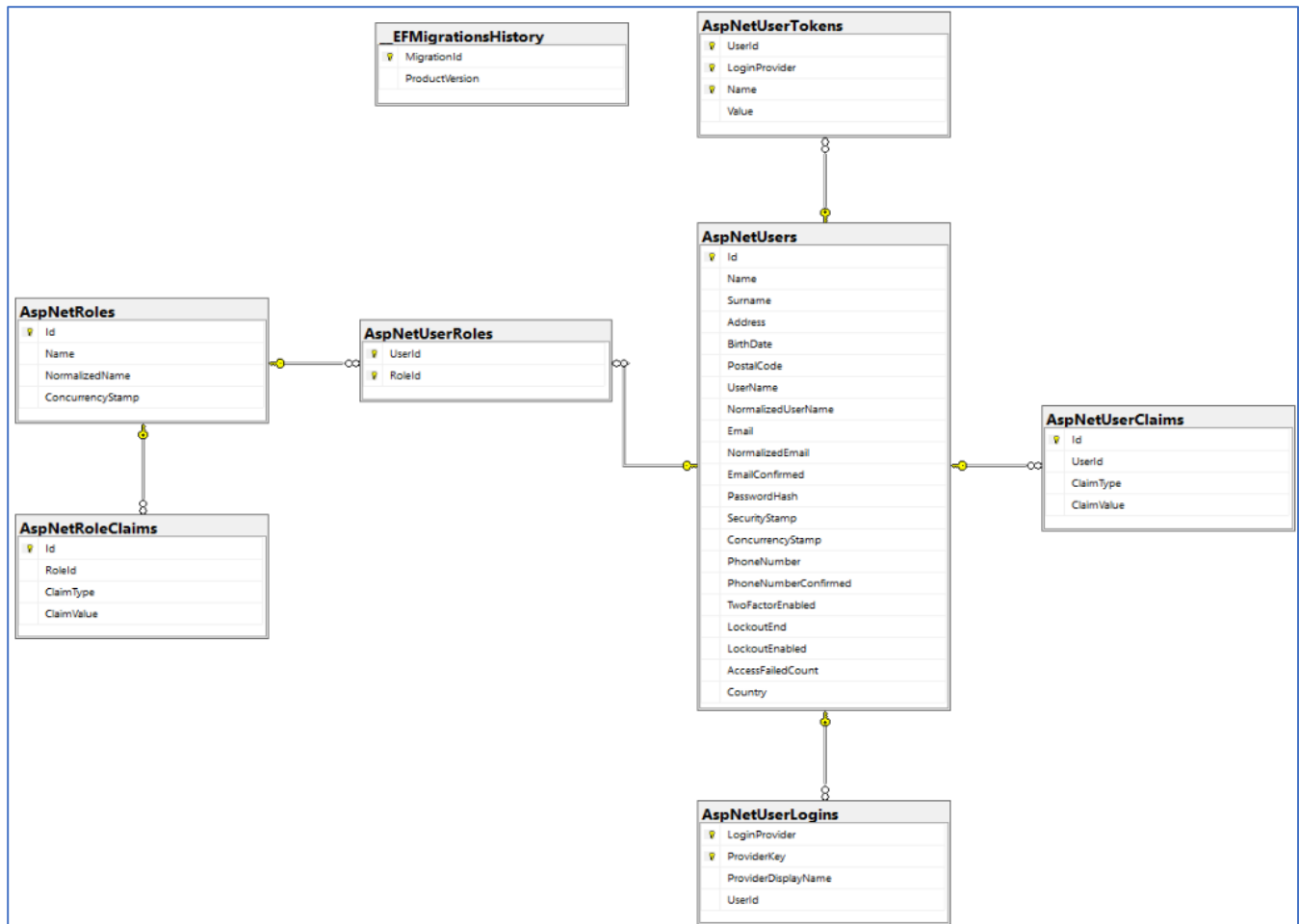
Entities



Capa de datos

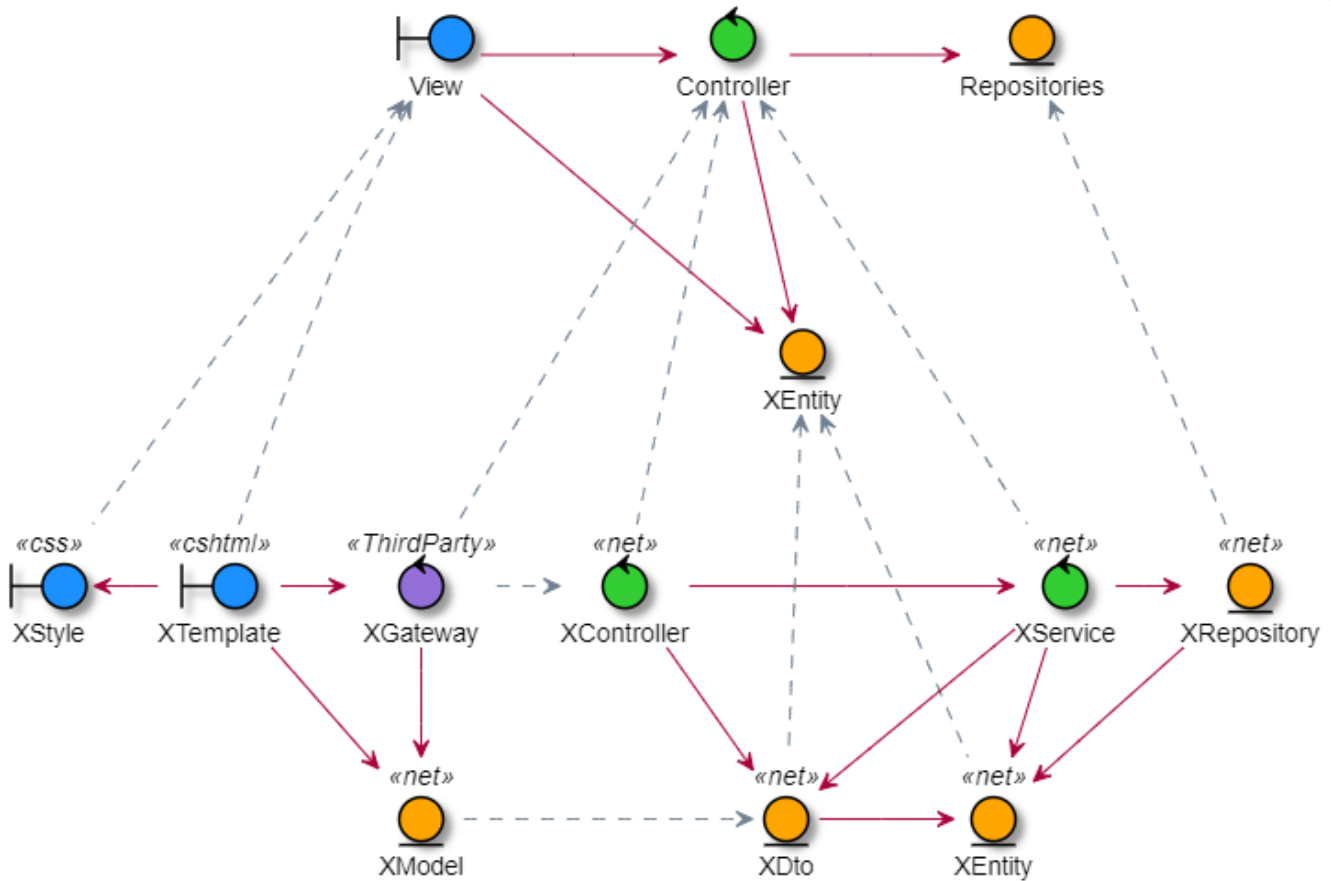
En este apartado se presenta el modelo de datos relación (E-R) que se ha realizado para representar mediante entidades los objetos descritos en los puntos anteriores, así como la relación que existe entre ellos.





Diseño de casos de uso

En este apartado se identificarán las clases de análisis necesarias para realizar los casos de uso y se trazaran sus nombres, responsabilidades y relaciones en un Diagrama de Clases. Además, se describirán las interacciones entre los objetos del análisis.



Implementación

En este apartado se ha dividido en nueve (9) partes, ecosistema de desarrollo, calidad de software, fiabilidad, mantenibilidad, seguridad, tamaño, duplicidad, complejidad y cobertura.

Ecosistema de desarrollo

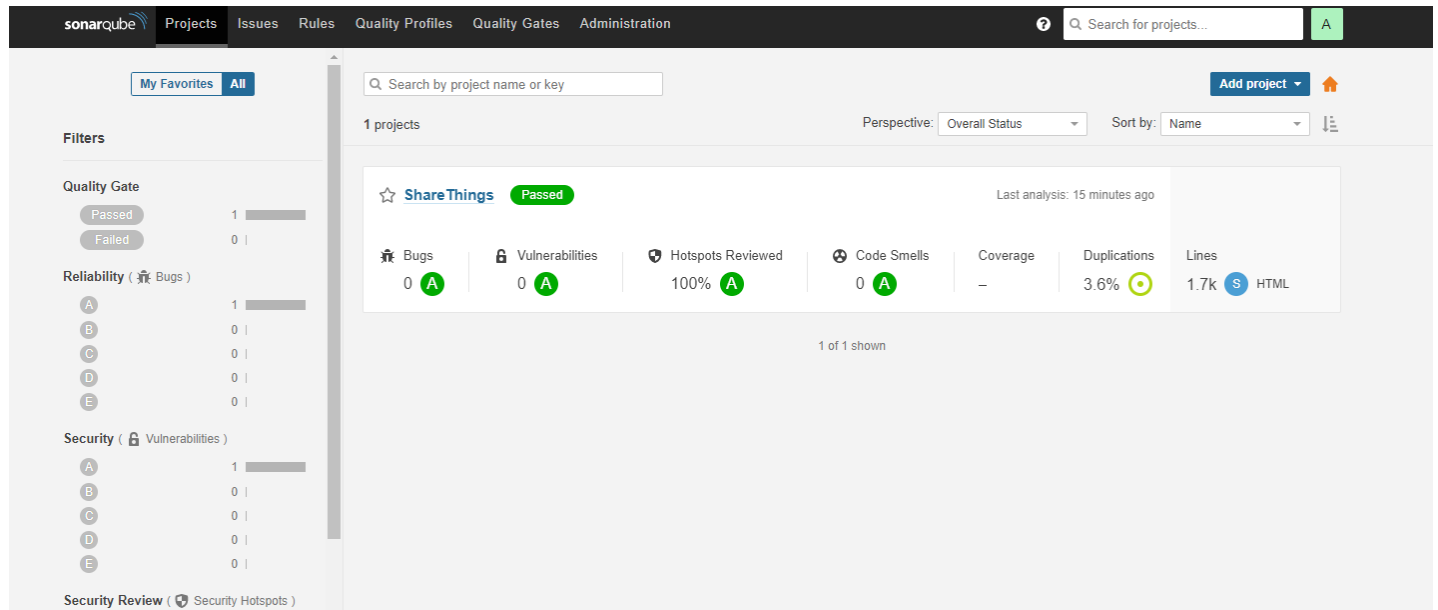
Se ha definido un ecosistema de desarrollo, el cual está formado por las siguientes herramientas:

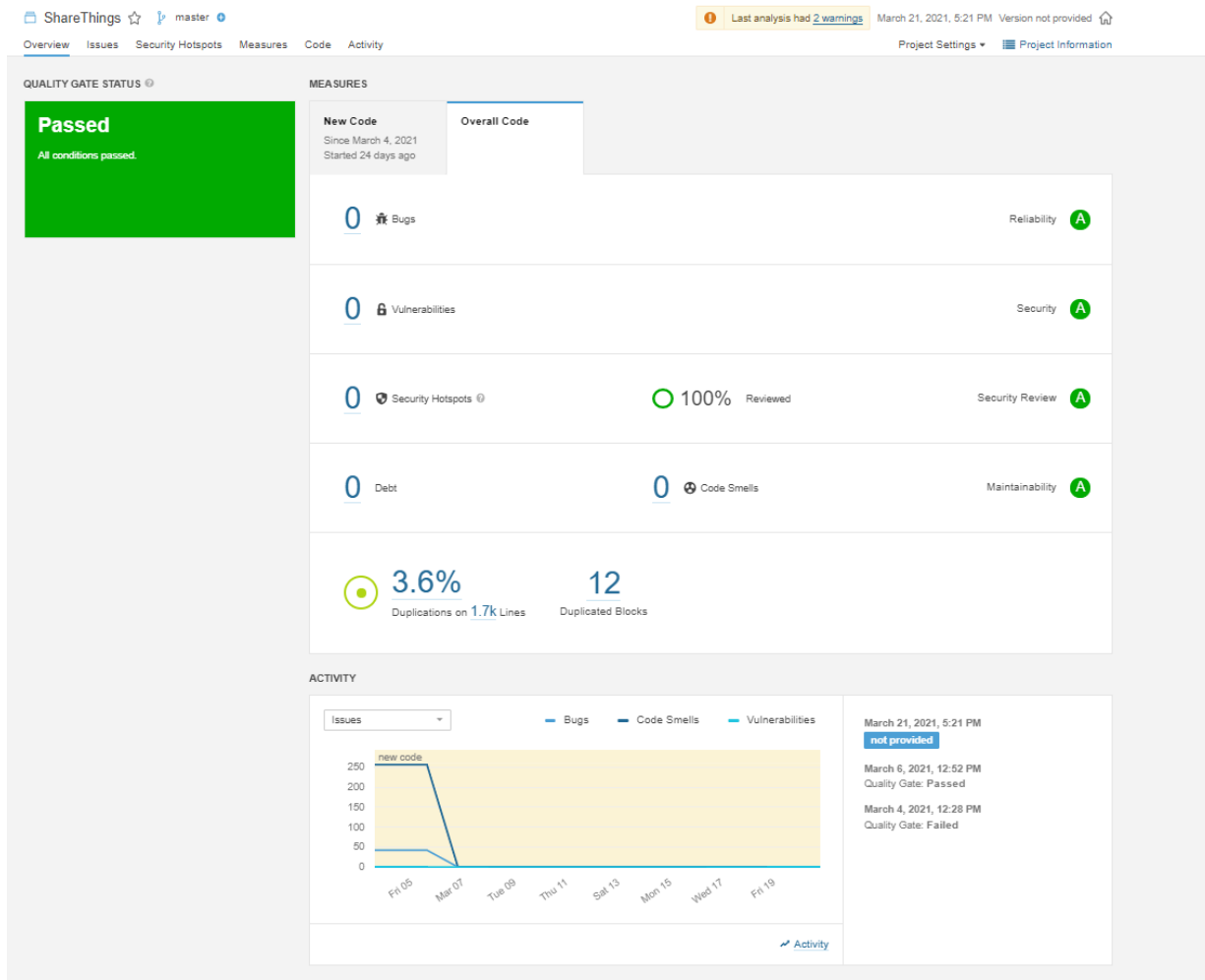
- BackEnd Framework: .Net 5
- FrontEnd Framework:
 - Razor
 - BootStrap
 - Javascript
- BBDD: SQL Server 2019
- ORM: Entitty Framework Core
 - CodeFisrt Mode
- Authorization management: Identity service
- Source Control: GitHub
- Testing Framework:
 - xUnit
 - Moq
 - TestHostServer
- Quality tool: SonarQube
- Continuous Integration: GitHub Actions
- Continuous Deployment: GitHub Actions
- Cloud Infrastructure: Azure
 - Blob Storage
 - App Service
 - SQL Database service
- Local infrastructure:
 - Docker Compose
 - IIS Express
- Third party tools: Sendgrid

Calidad del software

Para medir y obtener toda información referente al software implementado hemos usado la herramienta de análisis de código estático SonarQube. Hemos creado una instancia de SonarQube en Azure a partir de la siguiente plantilla ([link](#))

A continuación, se muestran las métricas obtenidas referentes al proyecto.





Se reducen los bugs, code smells y vulnerabilidades gracias a un trabajo sobre el código después de analizar los primeros resultados obtenidos con SonarQube. Y se realiza una configuración más realista de la herramienta en la que se desactiva el análisis de librerías third party de javascript como jQuery.

Configuration page for SonarQube, showing the 'Source File Exclusions' section.

Source File Exclusions

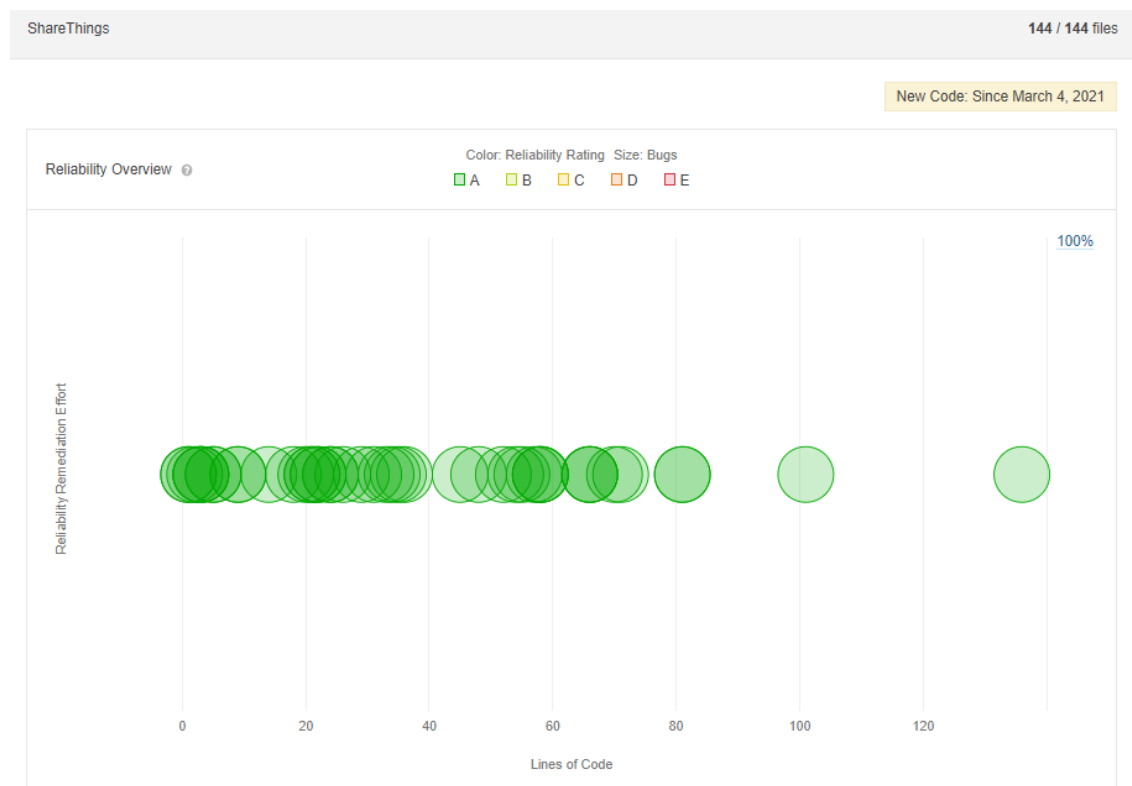
Patterns used to exclude some source files from analysis.

Key: sonar.exclusions

Excluded patterns:

- **/*bootstrap*
- **/*jquery*

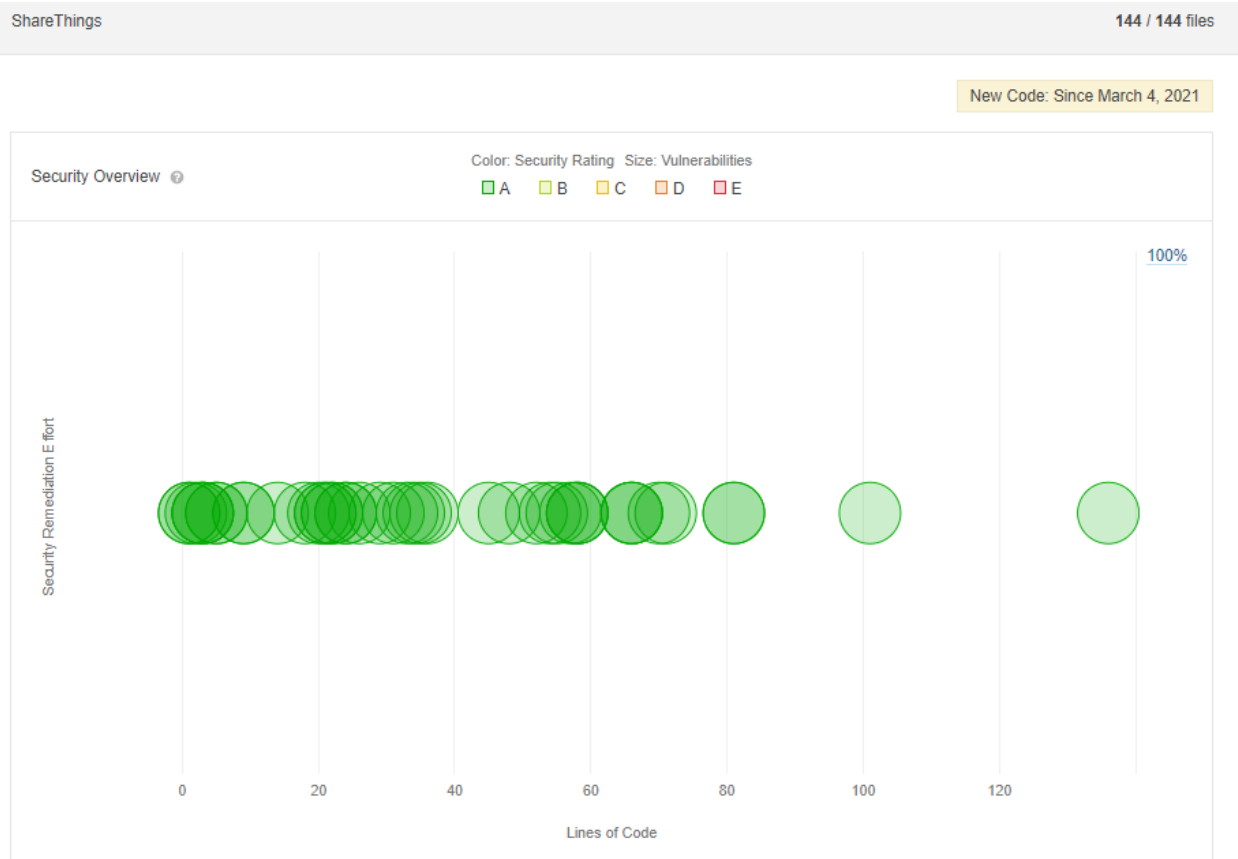
Fiabilidad



Mantenibilidad



Seguridad



Tamaño

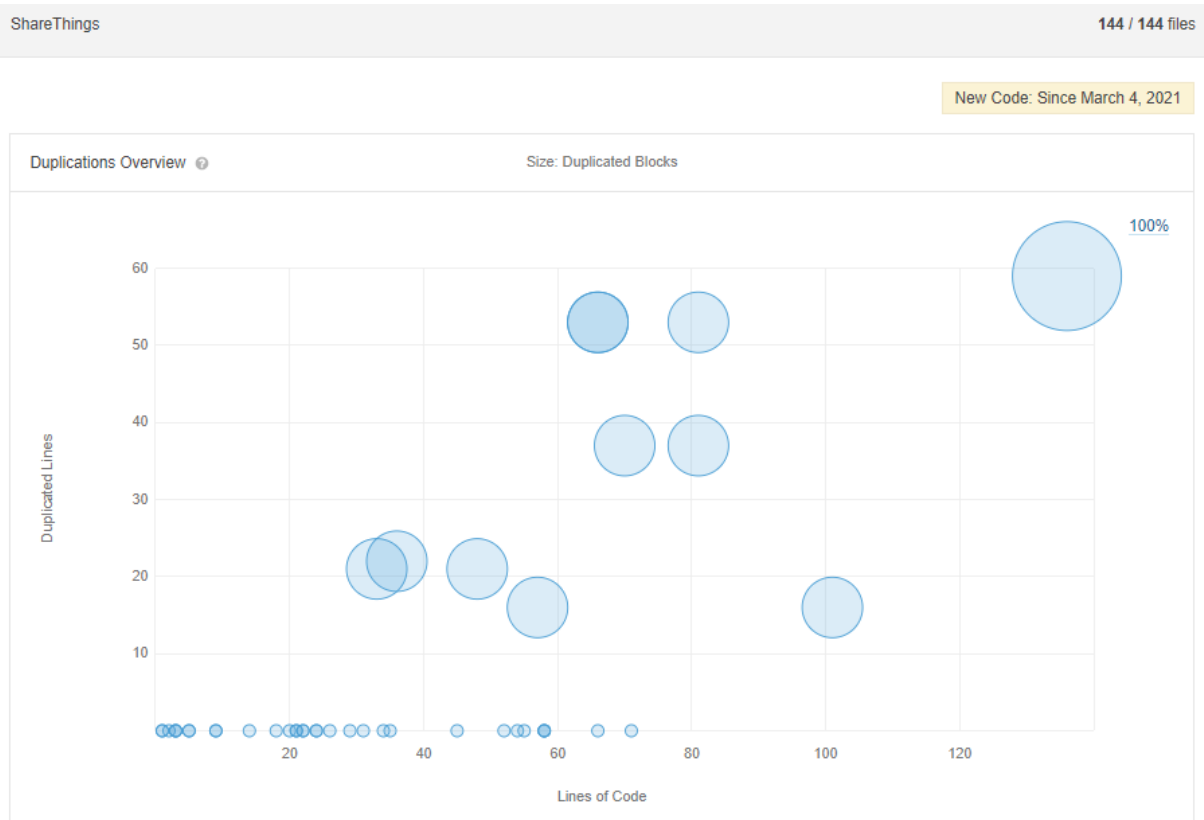
ShareThings View as Tree ↑ ↓ to select files ← → to navigate 5 files

New Lines **1,513** New Code: Since March 4, 2021

ShareThings	54
ShareThings.Data	402
ShareThings.Domain	15
ShareThings.FunctionalTest	780
ShareThings.UnitTest	262

5 of 5 shown

Duplicidades



Se extrae como indicador que se podría eliminar las duplicidades de código añadiendo componentes de vista que unifican el diseño en un solo elemento común a distribuir en las distintas páginas que lo requieran

ShareThings View as List ↑ ↓ to select files ← → to navigate 144 files

Duplicated Lines (%) 3.6%

New Code: Since March 4, 2021

	Duplicated Lines (%)	Duplicated Lines
ShareThings/Views/Products/Delete.cshtml	74.6%	53
ShareThings/Views/Products/StatusShary.cshtml	74.6%	53
ShareThings/Views/Products/Details.cshtml	62.4%	53
ShareThings/Views/Borrows/Reject.cshtml	51.2%	21
ShareThings/Views/Products/Create.cshtml	48.7%	37
ShareThings/Views/Borrows/Confirm.cshtml	46.8%	22
ShareThings/Views/Products/Edit.cshtml	42.5%	37
ShareThings/Views/Borrows/Index.cshtml	40.7%	59
ShareThings/Views/Borrows/Edit.cshtml	37.5%	21
ShareThings/Views/Borrows/OwnerDetail.cshtml	25.8%	16
ShareThings/Views/Borrows/_BorrowDetail.cshtml	15.0%	16

There are 133 hidden components with a score of 0.0%. [Show Them](#)

Cobertura de código

Para realizar el seguimiento de la cobertura de código hemos utilizado **Fine Code Coverage** del siguiente [enlace](#).

Fine Code Coverage

Coverage

Summary

Risk Hotspots

Rate & Review

Log Issue/Suggestion

Buy me a coffee

Collapse all | Expand all

Filter:

▼ Name	▼ Covered	▼ Uncovered	▼ Coverable	▼ Total		▼ Line coverage
+ ShareThings	805	581	1386	2780	58%	<div></div>
+ ShareThings.Data	182	0	182	334	100%	<div></div>
+ ShareThings.Domain	265	15	280	509	94.6%	<div></div>
+ ShareThings.FunctionalTest	497	2	499	982	99.5%	<div></div>
+ ShareThings.UnitTest	847	0	847	1329	100%	<div></div>
+ ShareThings.Views	382	114	496	1786	77%	<div></div>

Fine Code Coverage	
Coverage	Summary
Rate & Review	
Log Issue/Suggestion	
Buy me a coffee	
Assemblies:	6
Classes:	121
Files:	120
Covered lines:	2978
Uncovered lines:	714
Coverable lines:	3692
Total lines:	7678
Line coverage:	80.6% (2978 of 3692)

Pruebas

En este proyecto hemos implementado dos tipos de pruebas, que son:

- Pruebas unitarias
- Pruebas de integración

El objetivo de realizar las pruebas en el proyecto es asegurar una buena cobertura de calidad en el código desarrollado, y que el producto que estamos implementando tenga la calidad y funcionamiento esperado. Así mismo, las pruebas permiten obtener un feedback rápido del impacto de cada cambio durante los tiempos de desarrollo.

El modelo de pruebas realizado es el siguiente:

- Pruebas Unitarias sobre el modelo y los servicios de negocio para verificar su correcto funcionamiento.
- Pruebas Integración, desde el controlador hasta la capa de persistencia, permitiendo verificar que los diferentes módulos y/o servicios usados están en armonía cuando trabajan en conjunto.
- En la capa de presentación no se considera oportuno la realización de pruebas dada la baja complejidad de las interfaces de usuario. Además, la parte funcional ha sido cubierta en las pruebas de integración.

El total de pruebas se ha realizado con los siguientes frameworks:

- Pruebas Unitarias: XUnit, además han sido respaldadas con Moq para realizar el mockeo de los objetos necesarios.
- Pruebas Integración: XUnit, acompañado de TestHostServer para levantar un host en local y poder realizar llamadas http, EntityFrameworkMemory para poder levantar la persistencia en memoria sin necesidad de apuntar a una base de datos real.

Prueba	Duración	Rasgos	M
ShareThings.FunctionalTest (15)	10,7 s		
ShareThings.FunctionalTest.Controller (15)	10,7 s		
BorrowControllerFunctionalTest (1)	2,7 s		
GetControllerFunctionalTest (12)	3 s		
HomeControllerFunctionalTest (1)	2,7 s		
ProductControllerFunctionalTest (1)	2,3 s		
ShareThings.UnitTest (74)	558 ms		
ShareThings.UnitTest.Domain (17)	51 ms		
BorrowDomainTest (8)	17 ms		
ProductDomainTest (6)	15 ms		
TermDomainTest (2)	8 ms		
UserDomainTest (1)	11 ms		
ShareThings.UnitTest.Services (57)	507 ms		
BorrowServiceTest (37)	206 ms		
PhotoServiceTest (3)	141 ms		
ProductServiceTest (17)	160 ms		

Evaluación de las pruebas

En las pruebas unitarias no encontramos mayor dificultad que la de plantear los tests negativos. En cambio, en las pruebas de integración nos encontramos con los siguientes problemas:

Problema	Solución
Simulación de llamadas HTTP	Framework TestHostServer
Base de datos Cloud	Base de datos en memoria con EF Memory
Autenticación con distintos usuarios	Generación de autenticadores por cada uno de los perfiles dentro de la aplicación
Seguridad a nivel de CSRF en los métodos POST, PUT y DELETE de los formularios	Recuperación del token de los formularios mediante llamadas GET de carga de formulario.

Tener todas estas pruebas nos permite tener una gran red de seguridad, permitieron identificar errores cometidos en los desarrollos que realizábamos antes ni si quiera de proteger dichos cambios, evitando así romper la construcción de la aplicación en el Continuous Integration.

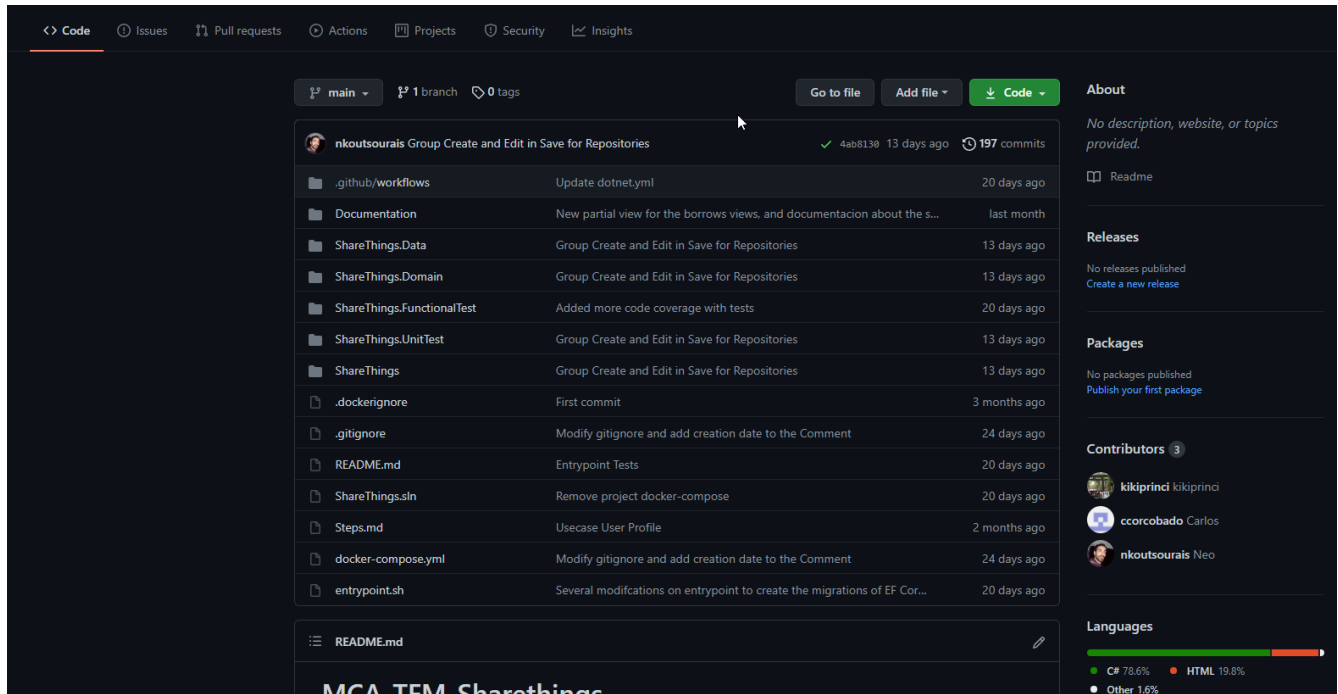
El conjunto de pruebas incluido en la aplicación se ejecuta también en el despliegue como posteriormente se verá.

Despliegue





























En este apartado hemos definido cinco (5) partes, control de versiones, entorno Cloud, despliegue, diagrama de hardware y entorno de desarrollo.

Control de versiones

Para este apartado hemos elegido la plataforma de GitHub por el conjunto de herramientas que nos proporciona para trabajar en grupo y de gestionar integración (CI) y entrega continua (CD).



<https://github.com/MasterCloudApps-Projects/RUP-Arquitecturas-Agiles>

Add FunctionalTests nkoutsourais committed on 28 Feb	 dc2ae6f	
Added TODO Pantalla to BorrowViewModel ccorobado committed on 28 Feb ✓	 c76eaa3	
Add TODO to BorrowViewModel ccorobado committed on 28 Feb ✓	 a58ccb4	
Added BorrowServiceTest part 1 ccorobado committed on 28 Feb ✓	 a5a577f	
Merge branch 'main' of https://github.com/kikiprinci/MCA-TFM-Sharethings ... kikiprinci committed on 28 Feb ✓	 6c4551d	
Comment Feature kikiprinci committed on 28 Feb	 bb79c77	
Create FakeBuilder ccorobado committed on 28 Feb ✓	 4f16e67	
Merge branch 'main' of https://github.com/kikiprinci/MCA-TFM-Sharethings ... kikiprinci committed on 28 Feb	 e9c9837	
Borrow List kikiprinci committed on 28 Feb	 8d368ed	
Complete implementation for all borrowService methods ccorobado committed on 28 Feb ✓	 753dda2	
Merge solve kikiprinci committed on 28 Feb	 5264753	
Create borrow kikiprinci committed on 28 Feb	 928acd3	
Merge branch 'main' of https://github.com/kikiprinci/MCA-TFM-Sharethings ... nkoutsourais committed on 28 Feb ✓	 15fa516	
Add PhotoService nkoutsourais committed on 28 Feb	 ab88d83	

Entorno Cloud

Dado que se decidió realizar la implementación con tecnología Microsoft optamos por su nube, Azure con todas sus herramientas disponibles.

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes the Microsoft Azure logo and a search bar. The left sidebar contains various navigation options like 'Información general', 'Registro de actividad', 'Control de acceso (IAM)', 'Etiquetas', 'Diagnosticar y solucionar pro...', 'Seguridad', 'Eventos', 'Facturación', 'Facturas', 'Información sobre el partner', 'Configuración', 'Implementación mediante pr...', and 'Grupos de recursos'. The main content area displays the 'Azure para estudiantes' subscription page. It shows a list of resource groups with columns for 'Nombre' and 'Suscripción'. The resource groups listed are 'DefaultResourceGroup-WEU', 'sharethings-quality', 'sharethings-sql', 'sharethings-web', and 'tfm-monolitos', all associated with the 'Azure para estudiantes' subscription.

Los servicios cloud utilizados fueron:

- **Sql Azure Database:** El cual está compuesto de un sql-server y un service plan.

The screenshot shows the Azure portal interface for the 'sharethings-sql' resource group. The left sidebar is the same as the previous screenshot. The main content area displays the 'sharethings-sql' resource group details. It shows the subscription 'Azure para estudiantes' and the subscription ID '3d98a042-577c-4c27-8fb0-fd7a0f626318'. Below this, there is a list of resources with columns for 'Nombre' and 'Tipo'. The resources listed are 'sharethings' (SQL Server), 'ShareThings (sharethings/ShareThings)' (Base de datos SQL), and 'sharethings-sql-plan' (Plan de App Service).

- **Blobstorage:** Sistema de almacenamiento para documentos. Se ha utilizado dos (2) contenedores para imágenes propias de la aplicación e imágenes asociadas a los productos que vienen proporcionadas por los usuarios.

The screenshot shows the Azure portal interface for the 'tfmmonolitos' storage account. The left sidebar is the same as the previous screenshots. The main content area displays the 'tfmmonolitos' storage account details. It shows a list of containers with columns for 'Nombre', 'Última modificación', and 'Nivel de acceso público'. The containers listed are 'basics' and 'photoproducts', both with a public access level of 'Blob'.

- **AppServices:** Para la parte de aplicaciones hemos usado dos servicios, uno donde tenemos un Sonarqube para análisis de calidad y otro para la propia aplicación de ShareThings, cada uno con su plan de coste separado e integrando una herramienta de logs como es Application Insights.
 - **sharethings-quality:** Grupo para la aplicación de Sonarqube.

sharethings-quality
Grupo de recursos

Buscar (Ctrl+/) << + Agregar Editar columnas Eliminar grupo de recursos Actualizar Exportar a CSV Abrir consulta Comentarios Abrir en

Información general

- Registro de actividad
- Control de acceso (IAM)
- Etiquetas
- Eventos

Configuración

- Implementaciones
- Seguridad
- Directivas

Información esencial

Suscripción (cambiar) : [Azure para estudiantes](#) Implementaciones :
 Id. de suscripción : 3d98a042-577c-4c27-8fb0-fd7a0f626318 Ubicación :
 Etiquetas (cambiar) : [Haga clic aquí para agregar etiquetas.](#)

Filtrar por cualquier ca... Tipo == todo Ubicación == todo Agregar filtro

Mostrando de 1 a 2 de 2 registros. ☐ Mostrar tipos ocultos

<input type="checkbox"/> Nombre ↑↓	Tipo ↑↓
<input type="checkbox"/> sharethings-quality-plan	Plan de App Service
<input type="checkbox"/> sharethingsqualitywww	App Service

- **sharethings-web:** Grupo para la aplicación de ShareThings.

sharethings-web
Grupo de recursos

Buscar (Ctrl+/) << + Agregar Editar columnas Eliminar grupo de recursos Actualizar Exportar a CSV Abrir consulta Comentarios Abrir en d

Información general

- Registro de actividad
- Control de acceso (IAM)
- Etiquetas
- Eventos

Configuración

- Implementaciones
- Seguridad
- Directivas
- Propiedades

Información esencial

Suscripción (cambiar) : [Azure para estudiantes](#) Implementaciones :
 Id. de suscripción : 3d98a042-577c-4c27-8fb0-fd7a0f626318 Ubicación :
 Etiquetas (cambiar) : [Haga clic aquí para agregar etiquetas.](#)

Filtrar por cualquier ca... Tipo == todo Ubicación == todo Agregar filtro

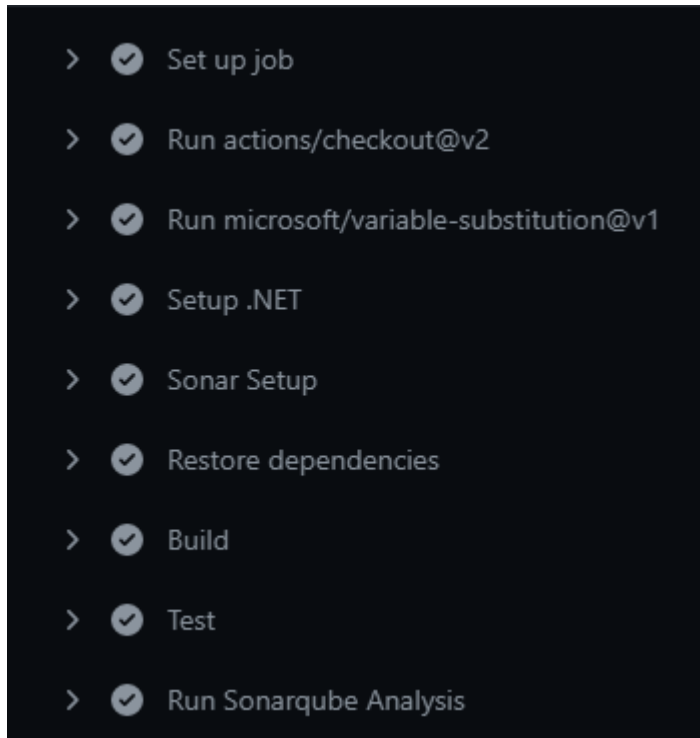
Mostrando de 1 a 3 de 3 registros. ☐ Mostrar tipos ocultos

<input type="checkbox"/> Nombre ↑↓	Tipo ↑↓
<input type="checkbox"/> sharethings	App Service
<input type="checkbox"/> sharethings	Application Insights
<input type="checkbox"/> sharethings-web-plan	Plan de App Service

Despliegue

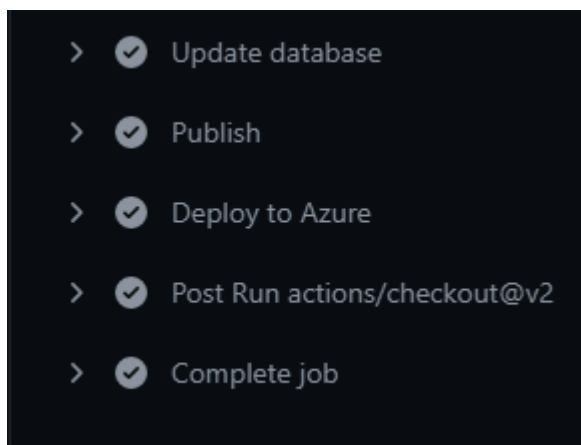
Como hemos mencionado en el apartado de control de versiones, Github nos proporciona un sistema para la gestión del CI y CD, llamado GitActions.

Este sistema funciona mediante la parametrización de pasos ejecutados secuencialmente, para la parte de integración continua (CI) estos son los pasos que hemos definido:



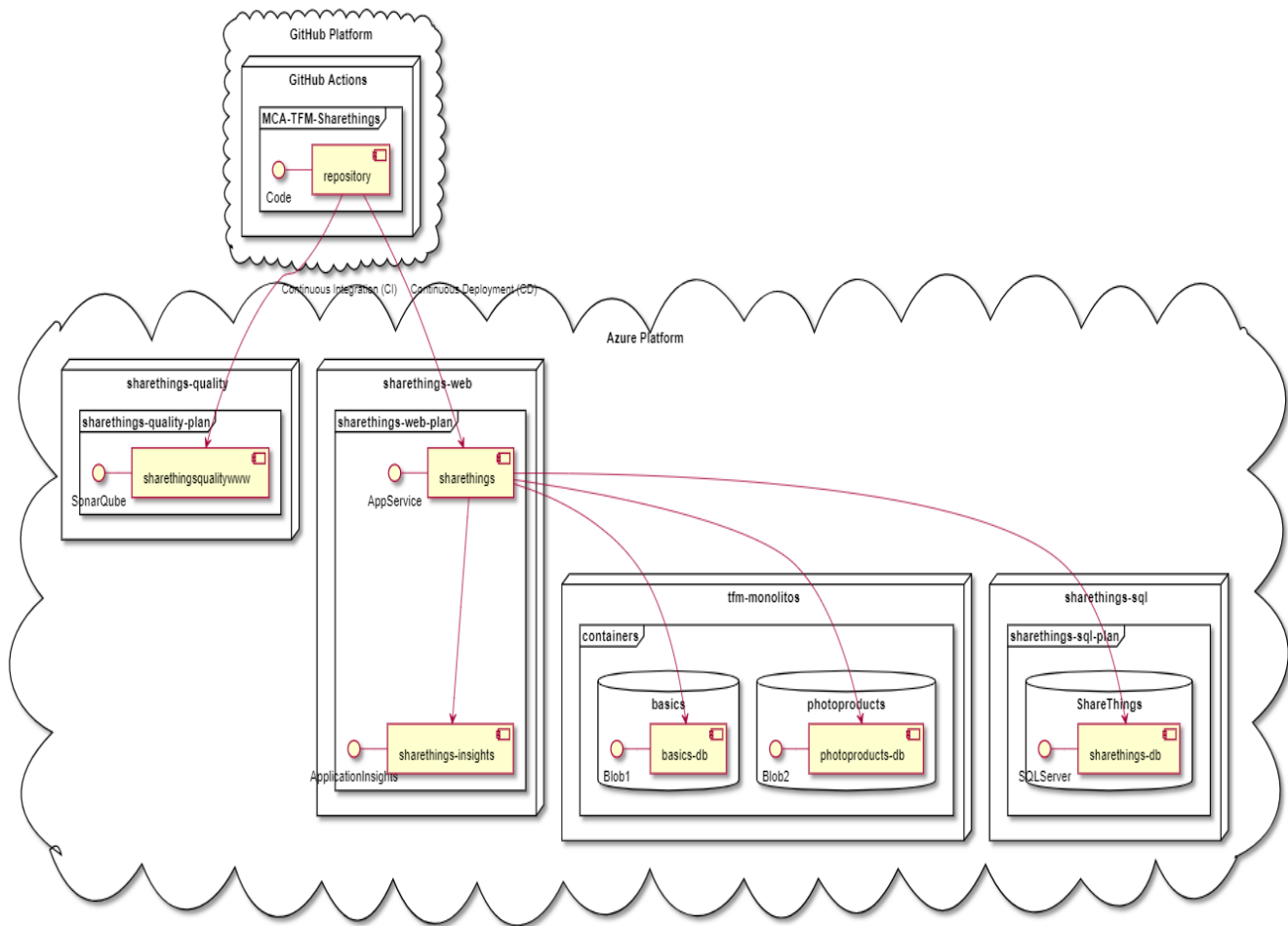
Para la sustitución de password del entorno producción se han utilizado las Secrets de Github.

En la parte de entrega continua (CD) los definidos son estos:



Cada ocurrencia está descrita en el fichero dotnet.yaml que contiene la aplicación dentro del directorio .github/workflows.

Diagrama de Hardware



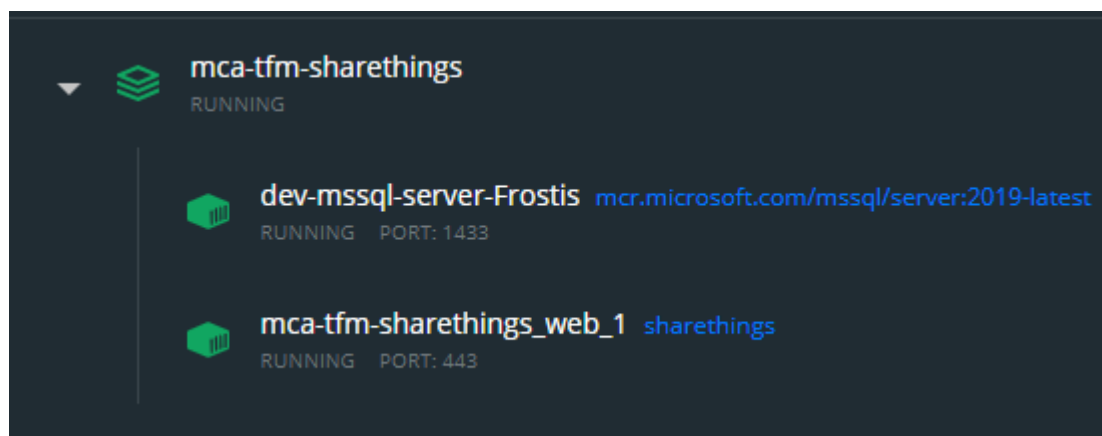
Entorno de desarrollo

Aprovechando los conocimientos aprendidos en el Máster Cloud Apps para el entorno local hemos utilizado toda la potencia que nos brinda Docker para no tener preinstalado nada en nuestros equipos locales pudiendo levantar el entorno completo como si fuera producción en contenedores.

Para esta tarea escribimos un dockerfile que levantaba la aplicación .Net y este estaba incluido dentro de un docker-compose que al mismo tiempo levantaba el sql-server.

```
version: '3.4'

services:
  db:
    image: mcr.microsoft.com/mssql/server:2019-latest
    container_name: dev-mssql-server-Frostis
    ports:
      - 1433:1433
    environment:
      SA_PASSWORD: 'P@ssw0rd!'
      ACCEPT_EULA: "Y"
  web:
    image: ${DOCKER_REGISTRY-}sharethings
    build:
      context: .
      dockerfile: ShareThings/Dockerfile
    environment:
      - ConnectionStrings__ShareThingsDatabase=Server=db;Database=ShareThings;User Id=sa;Password=P@ssw0rd!;
      # - ASPNETCORE_URLS=https://+:443;http://+:80
      - ASPNETCORE_Kestrel__Certificates__Default__Password=P@ssw0rd!
      - ASPNETCORE_Kestrel__Certificates__Default__Path=/app/ShareThings/Certs/ShareThings.pfx
    volumes:
      - ./ShareThings:/app/ShareThings:cached
      - ./ShareThings.Data:/app/ShareThings.Data:cached
      - ./ShareThings.Domain:/app/ShareThings.Domain:cached
      - ./entrypoint.sh:/entrypoint.sh
    working_dir: /app/ShareThings
    ports:
      - 443:443
      - 80:80
    entrypoint:
      - sh
      - /entrypoint.sh
    depends_on:
      - db
```



Arquitecturas A-giles y tal y tal ...

Para el desarrollo de este apartado se han analizado tres (3) patrones arquitectónicos muy utilizados en estos tiempos, que son los siguientes:

- Onion, de **Jeffrey Palermo**
- Hexagonal o Ports and Adapters, de **Alistair Cockburn**
- Clean Architecture, de **Robert C. Martin** (Uncle Bob)

Objetivos

El objetivo del análisis de estos patrones arquitectónicos es comparar las similitudes y diferencias que hay entre ellos. Además, se revisará por qué son patrones tan demandados en la actualidad y que impacto han tenido dentro de la industria del software.

Definiciones

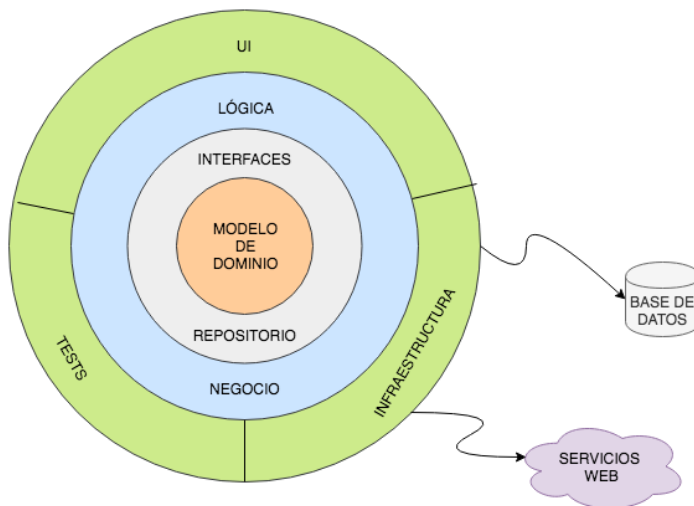
Onion

Se trata de un patrón arquitectónico multicapa que se desarrolla en torno al dominio independizando el resto de las capas. Las dependencias se resuelven de fuera hacia dentro, siendo el dominio el núcleo central.

Alrededor del dominio se define el comportamiento del almacenamiento de datos (repositorio) a través del uso de interfaces.

Por encima de estas se encuentra la capa de lógica de negocio que hace uso de estas interfaces de repositorio con su correspondiente implementación.

En las capas superiores se encuentran todos los detalles de comunicación con el exterior. Es en este punto donde se realizan las implementaciones de las interfaces de las capas interiores. Este comportamiento promueve la inversión de control de dependencias, la facilidad en realizar las pruebas de integración, así como la independencia de tecnologías, bases de datos y agentes externos promoviendo que quede intacta la lógica empresarial frente a los cambios.



Hexagonal o Puertos y adaptadores

Se trata de un patrón arquitectónico multicapa cuyo objetivo, una vez más, es poner en el centro el dominio y toda la lógica de negocio, así como el comportamiento del almacenamiento de datos o repositorio.

En este patrón arquitectónico se definen unas fronteras muy claras con el exterior, así se consigue que el core lógico no se corrompa con ningún agente externo, como acceso a datos, acceso a terceros e interacción con el usuario.

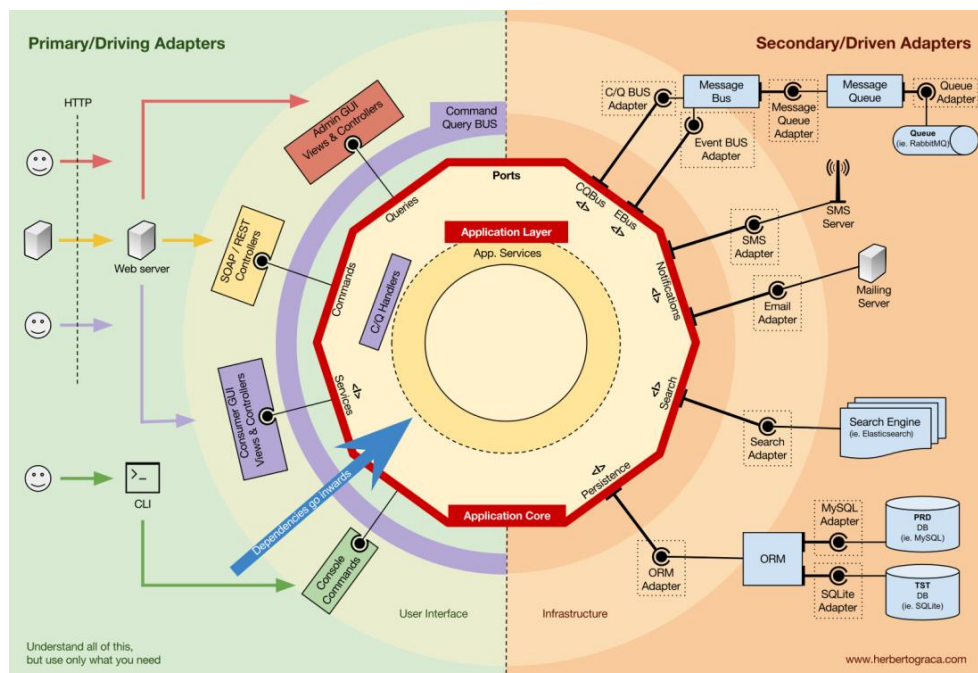
Existen tres (3) partes claramente diferenciadas, la lógica core del negocio, los puertos y los adaptadores.

Los puertos son las interfaces que definen la interacción con el exterior y exponen únicamente datos de nuestro dominio sin contaminar el interior.

Los adaptadores son los mecanismos usados para conectar el exterior con los puertos. Contienen la implementación de acceso a datos y acceso a recursos de terceros.

Los puertos y adaptadores se pueden dividir en primarios y secundarios, también llamados *driving* y *driven* respectivamente. Los primarios definen *qué se expone de nuestro sistema* al exterior, mientras que los secundarios definen *qué necesita nuestro sistema* del exterior. De una manera más genérica se podría decir que los primarios son entradas y los secundarios son salidas.

Este comportamiento promueve la inversión de control de dependencias, la facilidad en realizar las pruebas de integración, así como la independencia de tecnologías, bases de datos y agentes externos promoviendo que quede intacta la lógica empresarial frente a los cambios.



Clean architecture

Este patrón arquitectónico es una visión conjunta entre *Onion*, *Puertos y adaptadores* y *BCE* (que no se menciona en este documento), añadiendo nuevos términos propuestos por Uncle Bob. Su objetivo fue la unificación de todos estos patrones en uno único.

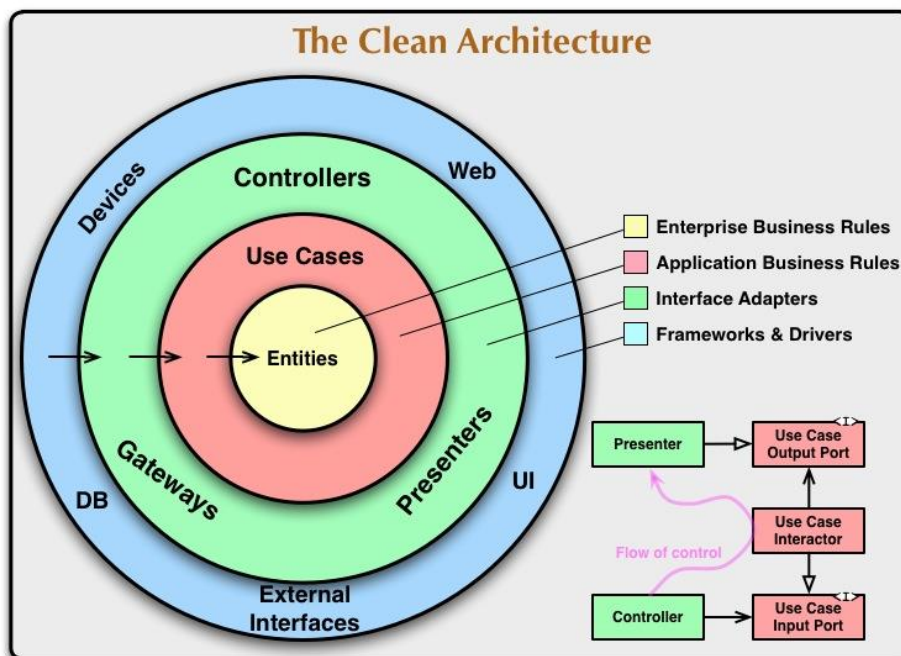
El objetivo de este patrón es el mismo que en los anteriores, lo cual promueve la inversión de control de dependencias, la facilidad en realizar las pruebas de integración, así como la independencia de tecnologías, bases de datos y agentes externos promoviendo que quede intacta la lógica empresarial frente a los cambios.

Este patrón se divide en cuatro (4) capas, donde tenemos de nuevo el dominio en el centro, aquí llamado entidades.

En la capa superior se encuentran los casos de usos, que definen la lógica de negocio de la aplicación. Aquí además están los puertos de entrada y salida similar a *Hexagonal*, así como la implementación de los de entrada llamados *Interactors*.

En la capa inmediatamente superior se encuentran los adaptadores, de tipo controladores, presentadores, acceso a terceros, etc.

Y la última capa define los sistemas con los que nos comunicamos, como base de datos, interfaces de usuarios, webs, dispositivos, etc.



Beneficios y desventajas

Patron	Beneficios	Desventajas
<i>Onion</i>	Mantenible Testeable Adaptable Agilidad (menor Time to Market) Facil de seguir Guiado por casos de uso Se cumple fácilmente la regla de la dependencia de fuera hacia a dentro Independencia de frameworks Independencia de UI Independencia de BBDDs Independencia de third parties	Curva de aprendizaje alta Aumento considerable de las clases
<i>Hexagonal / Puertos y adaptadores</i>	Testeable Mantenible Facilidad de identificar deuda técnica Se cumple fácilmente la regla de la dependencia de fuera hacia a dentro Independencia de frameworks Independencia de UI Independencia de BBDDs Independencia de third parties	Nido de interfaces Curva de aprendizaje alta Aumento considerable de las clases
<i>Clean Architecture</i>	Testeable Mantenible Fácil de seguir Agilidad Portable Guiado por casos de uso Se cumple fácilmente la regla de la dependencia de fuera hacia a dentro Independencia de frameworks Independencia de UI Independencia de BBDDs Independencia de third parties	Aumento considerable de las clases Nido de interfaces Curva de aprendizaje alta

Arquitectura/Características	<i>Onion</i>	<i>Hexagonal / Puertos y adaptadores</i>	<i>Clean Architecture</i>
<i>BENEFICIOS</i>			
Mantenible	✓	✓	✓
Testeable	✓	✓	✓
Adaptable	✓	✓	✓
Agilidad (menor Time to Market)	✓	✓	✓
Guiado por casos de uso	✓	✗	✓
Se cumple fácilmente la regla de la dependencia de fuera hacia a dentro	✓	✓	✓
Independencia de frameworks	✓	✓	✓
Independencia de UI	✓	✓	✓
Independencia de BBDDs	✓	✓	✓
Independencia de third parties	✓	✓	✓
Facilidad de identificar el no cumplimiento del patrón	✓	✓	✓
Portable	✓	✓	✓
<i>DESVENTAJAS</i>			
Curva de aprendizaje alta	✓	✓	✓
Aumento considerable de las clases	✓	✓	✓
Nido de interfaces	✗	✓	✓

Resumen

Una vez analizados en detalle los patrones arquitectónicos se aprecia claramente que los tres (3) podrían ser el mismo, ya que los beneficios y desventajas son prácticamente similares. Los tres (3) buscan los mismos objetivos, como se observa en la tabla anterior, la inversión de dependencias, ser fácilmente testeables, mantenibles y con el negocio independiente de cualquier tecnología, siendo sus pocas diferencias agrupadas en cambios en la nomenclatura de los elementos o en el uso de interfaces. Entendemos por ello por qué en muchos recursos y foros técnicos a los tres (3) patrones se les identifica como uno solo.

Existen diferentes opiniones en la industria del software sobre estos patrones con adeptos y detractores, para los primeros es un santo grial y para los segundos la reinención de la rueda.

Por todas las características anteriormente descritas se les considera arquitecturas ágiles, y esa es la razón por la que la industria ha adoptado fervientemente.

Después de analizarlas hemos sido conscientes de que realmente lo que hace que tengas todos los beneficios de los patrones anteriores es diseñar e implementar siguiendo los principios de diseño. La ayuda que proporcionan estas metas arquitecturas es una estructura y unas reglas que facilitan cumplir los principios de un buen diseño. Aunque en algunos casos se abuse del uso interfaces o en el aumento del número de clases, con la complejidad que ello conlleva.

Estos patrones arquitectónicos ayudan a democratizar los principios de diseño ya usados en el siglo pasado haciendo que la industria los adopte y facilite la resolución de problemas. Como consecuencia enmascaran ciertos principios o conceptos básicos de diseño y no promueven hacerse ciertas preguntas a quienes los usan, tales como el porqué de las cosas.

Según nuestro punto de vista, siempre que tengas los conceptos de diseño bien asentados, estas arquitecturas no serían tan necesarias ya que se puede desarrollar software de calidad que mantiene todos los beneficios descritos anteriormente.

Por tanto, según el prisma desde el que se oriente, se puede considerar que estos patrones son ventajosos o no. Si le das un enfoque a nivel empresarial tienes los beneficios de ser ágil partiendo de la base que el objetivo es maximizar el beneficio económico. Sin embargo, si le das un enfoque docente puedes perder el objetivo de hacer entender el concepto abstracto que está detrás de todos los principios que engloban estos patrones arquitectónicos.

Conclusiones

El objetivo de este TFM era poner en práctica todo lo aprendido durante el máster en las diferentes asignaturas y probar en nuestra propia piel una metodología que desconocíamos hasta el momento, como es Rational Unified Process (RUP), aunque no la vimos en detalle en el máster, Luis nos ha guiado con ella en este proceso.

Este TFM ha sido elaborado por tres personas, en un principio pensábamos ya que nos conocíamos del entorno laboral que esto nos supondría un menor esfuerzo dada el nivel de exigencia del trabajo sumado a los retos familiares de tener niñas pequeñas. Pero esto no fue así, los debates y el llegar a acuerdos no fue fácil, por lo que supuso un grado mayor dificultad, y un reto aún más apasionante.

Reconocemos que RUP nos ha sorprendido gratamente, no solo por lo que ya nos mencionó Luis hasta la saciedad, que el desarrollo sería mucho más fácil cuando tuviéramos TODO diagramado, sino porque rompió muchos silos y problemas de comunicación que teníamos, estableciendo un lenguaje común facilitándonos la comunicación y el establecimiento de conceptos generales en los flujos de trabajo. También queremos destacar la trazabilidad que nos ha permitido tener RUP sobre el proyecto y el desarrollo.

También nos ha permitido ampliar con más detalle principios y patrones de diseño y patrones arquitectónicos que oíamos nombrar mucho en el sector pero que no encontrábamos el tiempo necesario para poder abordar. Con este TFM salimos enriquecidos técnicamente y junto con el máster con el síndrome del impostor un poco más diluido.

Bibliografía

Anderson, Rick. *Enrutar a Acciones de Controlador de ASP.NET Core* | Microsoft Docs.

<https://docs.microsoft.com/es-es/aspnet/core/mvc/controllers/routing?view=aspnetcore-5.0>. Accessed 3 Apr. 2021.

---. *Introducción a Identity En ASP.Net Core* | Microsoft Docs. <http://docs.microsoft.com/es-es/aspnet/core/security/authentication/identity?view=aspnetcore-5.0&tabs=visual-studio>.

Accessed 3 Apr. 2021.

---. *Middleware de ASP.NET Core* | Microsoft Docs. <https://docs.microsoft.com/es-es/aspnet/core/fundamentals/middleware/?view=aspnetcore-5.0>.

Accessed 3 Apr. 2021.

---. *Scaffolding Identity En Proyectos de ASP.Net Core* | Microsoft Docs. <https://docs.microsoft.com/es-es/aspnet/core/security/authentication/scaffold-identity?view=aspnetcore-5.0&tabs=visual-studio>.

Accessed 3 Apr. 2021.

Ardalis. *Arquitecturas de Aplicaciones Web Comunes* | Microsoft Docs. <http://docs.microsoft.com/es-es/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures>.

Accessed 3 Apr. 2021.

---. *Información General de ASP.NET Core MVC* | Microsoft Docs. <http://docs.microsoft.com/es-es/aspnet/core/mvc/overview?view=aspnetcore-5.0>.

Accessed 3 Apr. 2021.

ASP NET Core Identity Tutorial - TekTutorialsHub. 27 Sept. 2017, <http://www.tektutorialshub.com/asp-net-core/asp-net-core-identity-tutorial/>.

ASP.NET Core 2.1.0-Preview1: Introducing Identity UI as a Library | ASP.NET Blog. 2 Mar. 2018,

<http://devblogs.microsoft.com/aspnet/aspnetcore-2-1-identity-ui/>.

ASP.NET Core MVC Request Life Cycle. <http://www.c-sharpcorner.com/article/asp-net-core-mvc-request-life-cycle/>. Accessed 3 Apr. 2021.

Etimbuk. *Dockerize Microsoft SQL Server on MacOS* | by Etimbuk U | Medium. Medium, 11 Feb. 2019,

<http://euedofia.medium.com/dockerize-microsoft-sql-server-on-macos-a8d40f8b1e66>.

Mark Otto, Jacob Thornton, and Bootstrap contributors. *Grid System · Bootstrap*.

<http://getbootstrap.com/docs/4.0/layout/grid/>. Accessed 3 Apr. 2021.

Murugan, Mukesh. *Custom User Management in ASP.NET Core MVC with Identity*.

<http://codewithmukesh.com/blog/user-management-in-aspnet-core-mvc/>. Accessed 3 Apr. 2021.

NAYAK, KESHAB. *ASP.NET Core: The MVC Request Life Cycle – Tech Blog*.

<https://www.techbloginterview.com/asp-net-core-the-mvc-request-life-cycle/>. Accessed 3 Apr. 2021.

Rodríguez, Kevin. *Enabling Email Verification in ASP.Net Core Identity UI 2.1* | by Kevin Rodríguez | Medium.

Medium, 12 June 2018, http://medium.com/@MisterKevin_js/enabling-email-verification-in-asp-net-core-identity-ui-2-1-b87f028a97e0.

Ryan. *ASP.NET Core C# - Send Email Messages with SendGrid API*. 18 Oct. 2020, <http://www.ryadel.com/en/asp-net-core-send-email-messages-sendgrid-api/>.

Cockburn, Alistair. Hexagonal Architecture – Alistair Cockburn. <https://alistair.cockburn.us/hexagonal-architecture/>. Accessed 11 Apr. 2021.

DDD, Hexagonal, Onion, Clean, CQRS, ... How I Put It All Together – @hgraca. 16 Nov. 2017, <https://herbertograca.com/2017/11/16/explicit-architecture-01-ddd-hexagonal-onion-clean-cqrs-how-i-put-it-all-together/>.

Martin, Robert C. Clean Architecture. Pearson Professional, 2018.

---. Clean Code. Pearson Education, 2009.

Palermo, Jeffrey. “Onion Architecture | Programming with Palermo.” Programming with Palermo, <https://jeffreypalermo.com/tag/onion-architecture/>. Accessed 11 Apr. 2021.

Reyes, Javier Velez. Javier Vélez Reyes. 20 Feb. 2016, <https://javiervelezreyes.com/ni-nueva-ni-arquitectura-ni-hexagonal/>.

Apuntes y documentación recibida en el Master Cloud Apps del curso 2019-2020