Ecole supérieure en informatique de sidi bel abbès

2021/2022

RÉALISATION D'UNE PLATEFORME COMMUNAUTAIRE DE LOCATION DE LOGEMENTS

Préparé et presenté par

Ilyes reda Merzouk / chef d'équipe
Abir Ben Aissa
Boulma Dacine Yousra
Feddag Mohammed Zineddine

Encadré par

Dr. Nadia Elouali

Table of Contents

General Introduction

I General Introduction

Most travelers primarily use hotels for accommodation, either for vacations or a short period of life for work or other , however lately a new promising concept of renting houses appeared and it’s making a big success . But how to choose your ranting service so as it provides you a trusted and cozy experience ?

Rented is a website that connects people looking for accommodation with people who are looking to rent their homes . It has multiple advantages comparing to the old methods because it makes you feel secure and comfortable.

The strong point of Rented is the safety and confidentiality , People who want to rent their homes will receive a check-up visit to confirm their offer .

Thus, we guarantee that the offer respects the description given on the web site and we insure all customers .

In the present report we are going to detail how Rented work and how it perfectly serve the customers needs .

project analysis

II project analysis

II.1 Study of the existing

For holidays , special events , journey or a home-stays getting a hotel room can be expressive and booking from random people on internet can be dangerous or the local is far from the client expectations (using fake photos or lying about the details of the rented place) .

II.2 Proposed solution

RENTED is an intermediate between the client and the host to ensure a better experience for both sides . It allows the client to find the desired lodging/room according to the place/city, price and other criteria.

Also it allows the host to rent their empty places (room , apartment...) and choose the client for whom to rent .

This online platform assign an agent to verify each posed post to avoid any fraud .

II.3 Specification of requirements

II.3.1 Functional requirements

The are major costumers for this application, one is the host who is the lodger ans the other is the client who wants to book from the host . Also we have the agent and the admin as an intermediate between them .

II.3.1.1 Host side

Add post to introduce the space to rent by specifying all the details . Allow the host to see , update and delete his posts see all the reservations and their states.

II.3.1.2 Client side

See all verified posts and interact with them (like , add to wish list , add comment ...) , search for posts using different filters ,suggest posts (recommendation) , make reservation and cancel it , claim a post for admin in case of conflicts .

II.3.1.3 Admin

See all posts and approve/decline them according to the agent feedback ,assign the verification mission an agent , approve users accounts according to their uploaded id , manage the conflicts between the host and the client .

II.3.1.4 Agent

Schedule an appointment to visit the rented space ,visit the local and make a feedback .

II.3.2 Non-functionals requirements

- **Low latency .**
- **High security .**
- **Data integration .**
- **High availability .**
- **High consistency .**
- **Security .**

II.4 Target Audience

Who RENTED is targeting with their business ?

RENTED is targeting travelers and hosts , they are both a vital for the business because they are the center of profitability .

II.4.1 The Client / travelers

Include mostly those who enjoy traveling and at the same time not spending all of their money on a hotel room .

II.4.2 Host

Involves owners or renters who are willing to rent their place (apartment , pool , villa , room ...) they might want to make money out of an unoccupied place or just want to meet interest people ,

Project Conception

III Project Conception

Defining in a none ambiguous way the functional needs and establishing the list of requirements, documenting and organizing them is a very important step that requires thorough preparation in order to avoid anomalies.

This chapter is devoted to the step 'Conception and Modeling Phase' using the modeling language UML (Unified Modeling Language) to provide a standardized method to visualize the conception and the architecture which show the different functionalities of our website.

III.1 The chosen modeling language UML

UML, also known as "Unified Modeling Language", is a tool that allows developers to present the design, documentation, requirements of a specific software, its creation was based on three methods: OMT , BOOCH, and OOSE. This language aims to simplify and facilitate the understanding of complicated systems. A graphical interface is generally used to exploit a design. Also, it helps developer teams to communicate and express their ideas in an obvious way (group work).

III.2 Project's main functionalities

This project aims to create a platform that is a two-sided marketplace, it seeks to match people that own real estate properties with people interested in renting short-term lodging.This procedure happens after several verifications and validations since security and reliability are guaranteed and a priority.

III.2.1 Needs specification

To express the specification of the needs, UML diagrams are set. There are several diagrams in the UML language which belong to different phases, according to different points of view. We will use some of them for the design of our web application.

III.2.1.1 Use case Diagram

The roles of use case diagrams are to collect, analyze and organize requirements, as well as identify the major functionalities of a system. Therefore, this is a very important step for the conception of our website.

III.2.1.1.1 Description of the General use case

Pre-condition: Authentication.

Main Actors:

- Client.
- Host.
- Agent.
- Admin.

Secondary Actor: /

Scenario:

- A client can accomplish a search for the posts. He can consult a post and send a request to rent. After sending that request, it can be canceled if it is not confirmed yet.

- A host can make, update, delete a post. He can also consult the list of posts and the posts calendar.

- An agent verifies a post.

- An admin validates or decline a post after the verification of the agent.

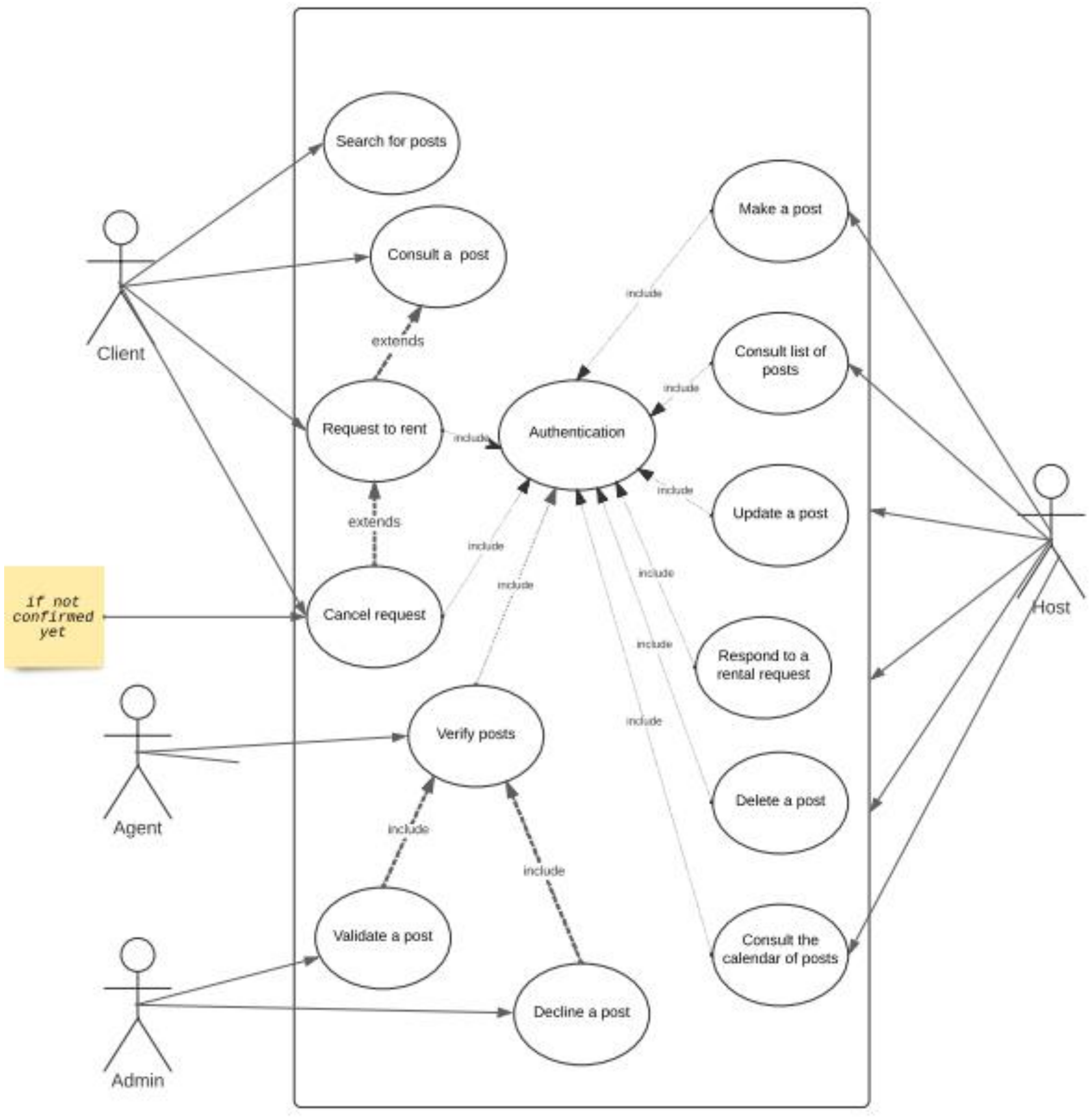


Figure 1 : General use case Diagram

III.2.1.1.2 Description of the accounts management use case diagram

Pre-condition: /

Main Actors:

- Client.
- Admin
- Host.

Secondary Actor: /

Scenario:

- A client can sign-up, after that, he has the ability to sign-in, reset his password, upgrade to host and sign-out. To update his account, his must be signed-in.
- An admin can view all the accounts, delete accounts and rest his own password, but to do that he must be signed-in. After signing-up, he has the ability to sign-out.
- To update his account, a host must be signed-up. After the sign-up, he can reset his password or sign-out.

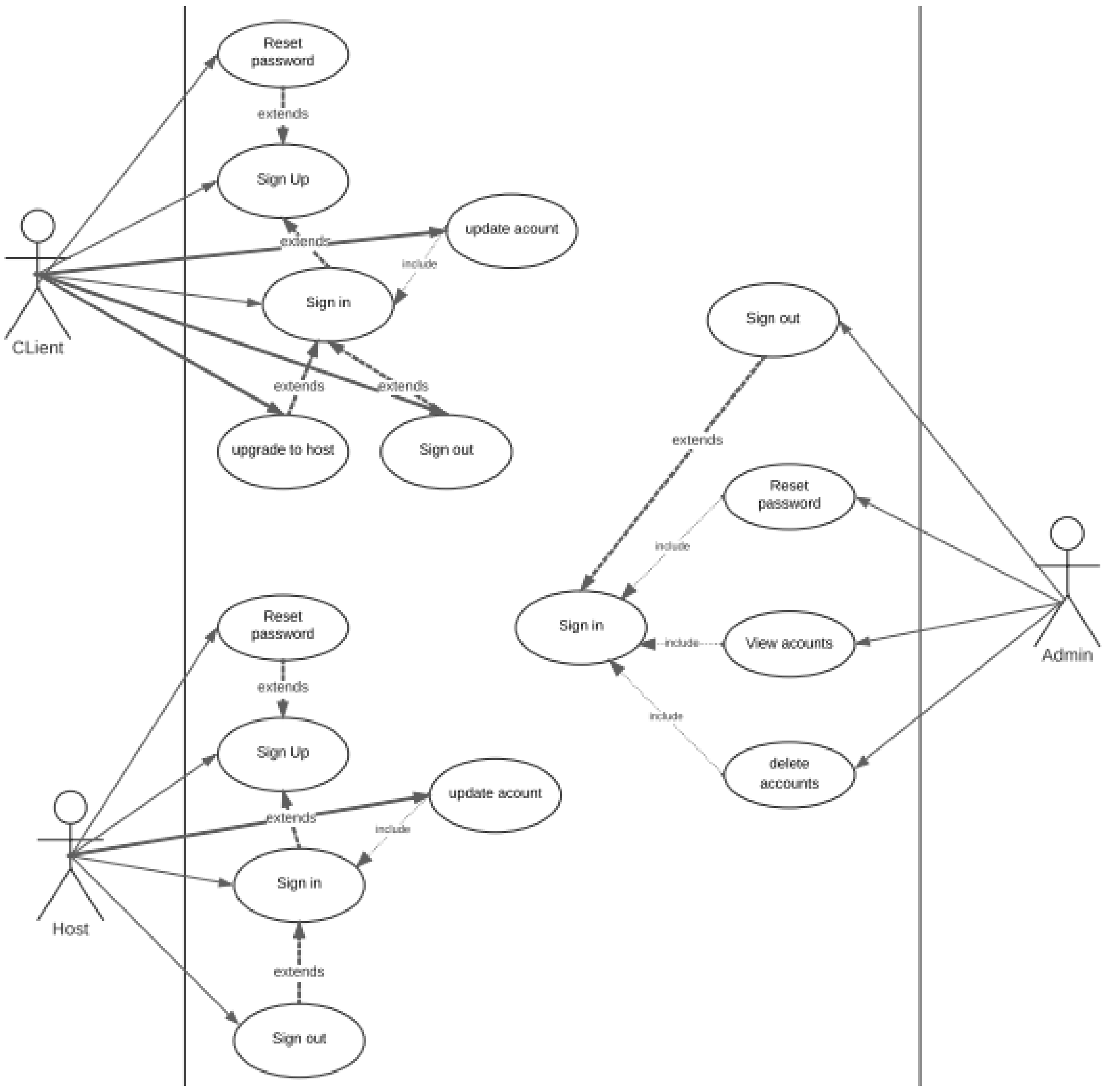


Figure 2 : Accounts management use case Diagram

III.2.1.1.3 Description of the reservation management use case diagram

Pre-condition:

- Authentication of the host.
- Authentication of the client.

Main Actors:

- Client.
- Host.

Secondary Actor: /

Scenario:

- A client can search for posts by accomplishing a filter depending on his preferences. He can consult a post and by doing that he can request this specific post for a rent. A cancel of the rent is also possible. Of course, to do all these actions, an authentication much be done first.
- A host can consult the list of posts and the posts calendar. Along with that, he can make, update and delete a post. After a request from a client for a rent, he must answer this rental request.

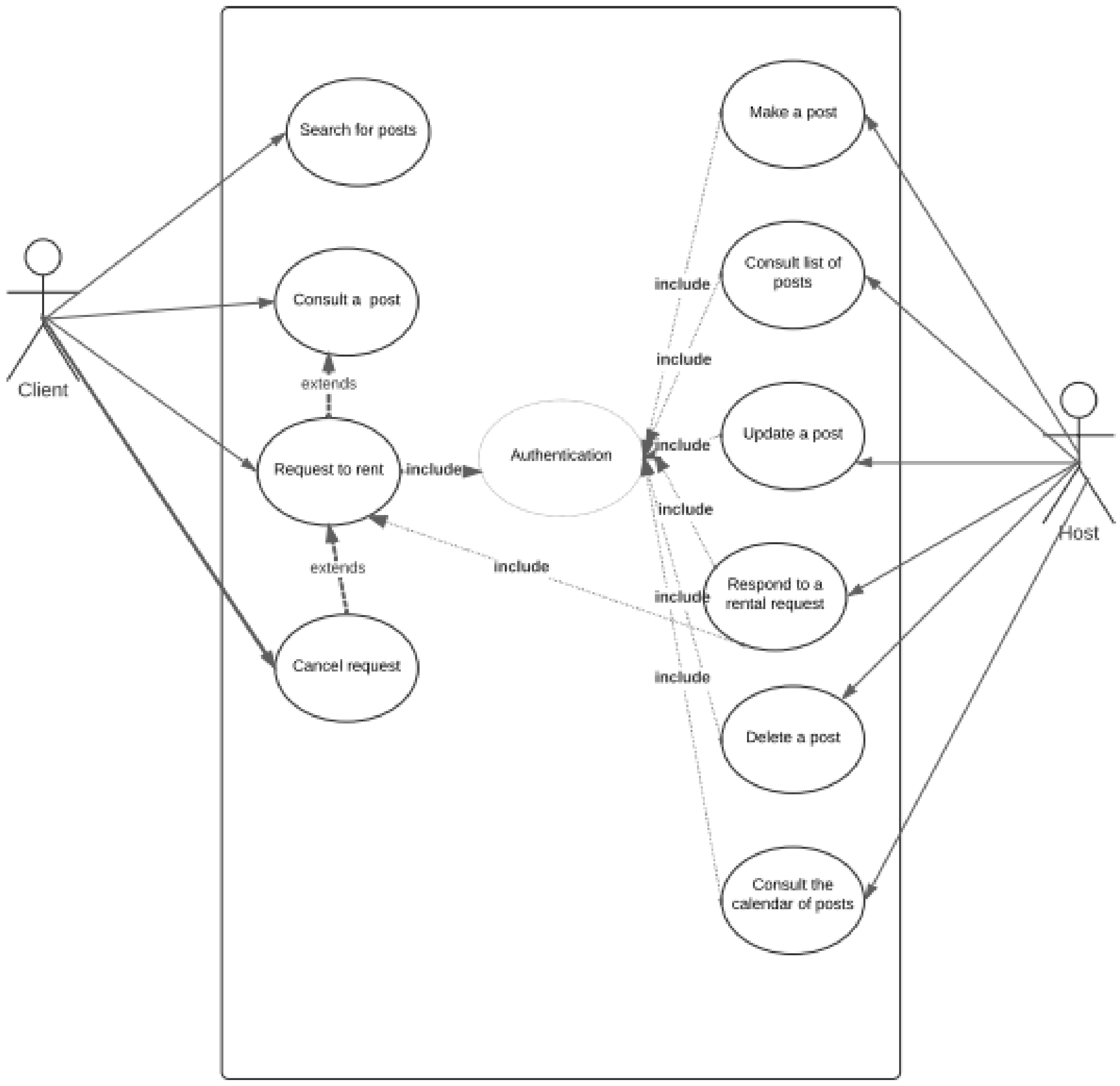


Figure 3 : Reservation management use case Diagram

III.2.1.1.4 Description of the verification post use case diagram

Pre-condition:

- Authentication of the Verification agent.
- Authentication of the admin.

Main Actors:

- Verification agent.
- Admin.

Secondary Actor: host

Scenario:

- The verification agent verifies the request of the host.
- The administrator either validates or declines the result of the agent's verification.

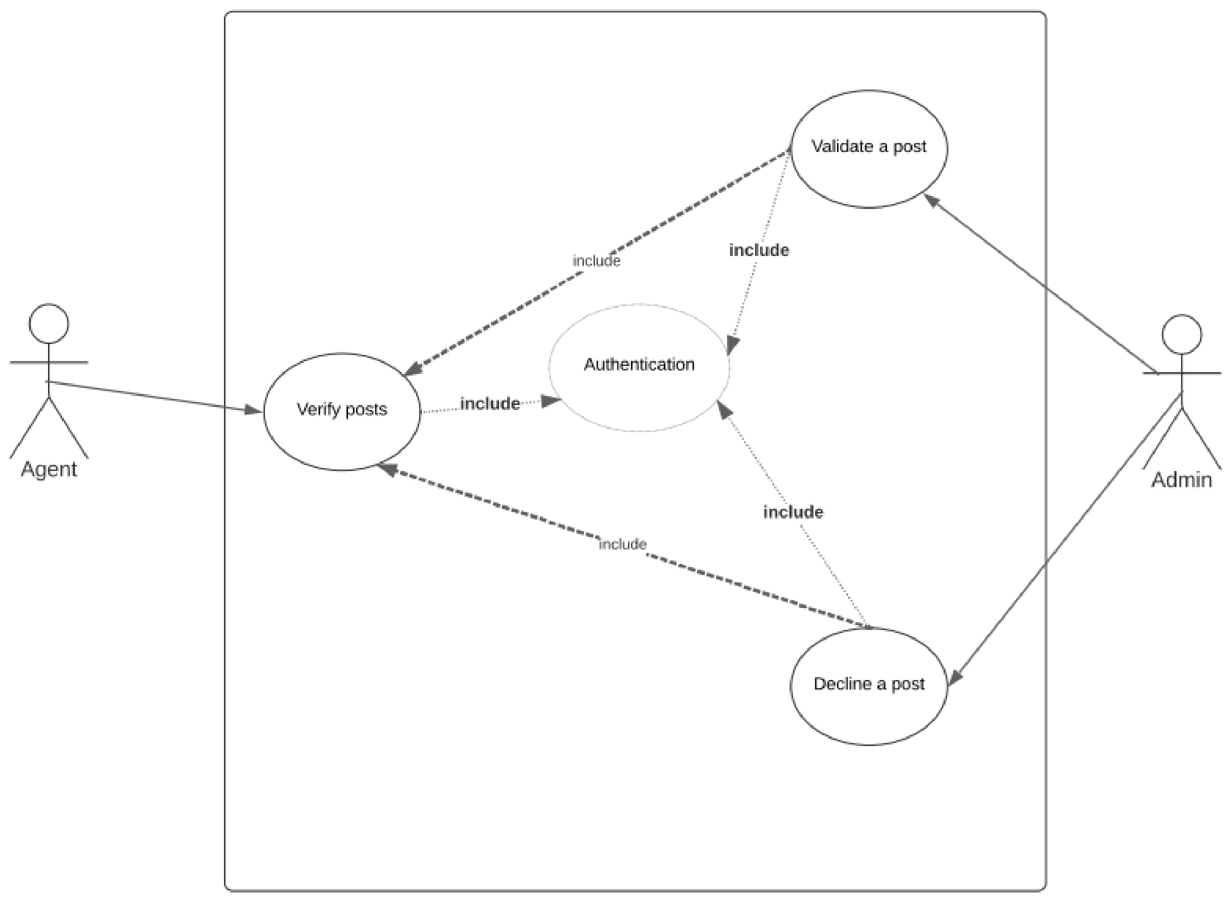


Figure 4 : Verification post use case Diagram

III.2.1.2 Sequence Diagram

A sequence diagram shows the sequence of messages passed between different objects. It can also show the control structures between those objects with lifelines of a specific scenario.

III.2.1.2.1 Description of the sign-up sequence diagram

Pre-condition: /

Main Actors:

- User

Secondary Actor:

- Administrator.

Scenario:

A sign-up of a user must be validated by the administrator. If it is validated, the user is informed by a notification, else, he is asked to retry and reinsert the needed information.

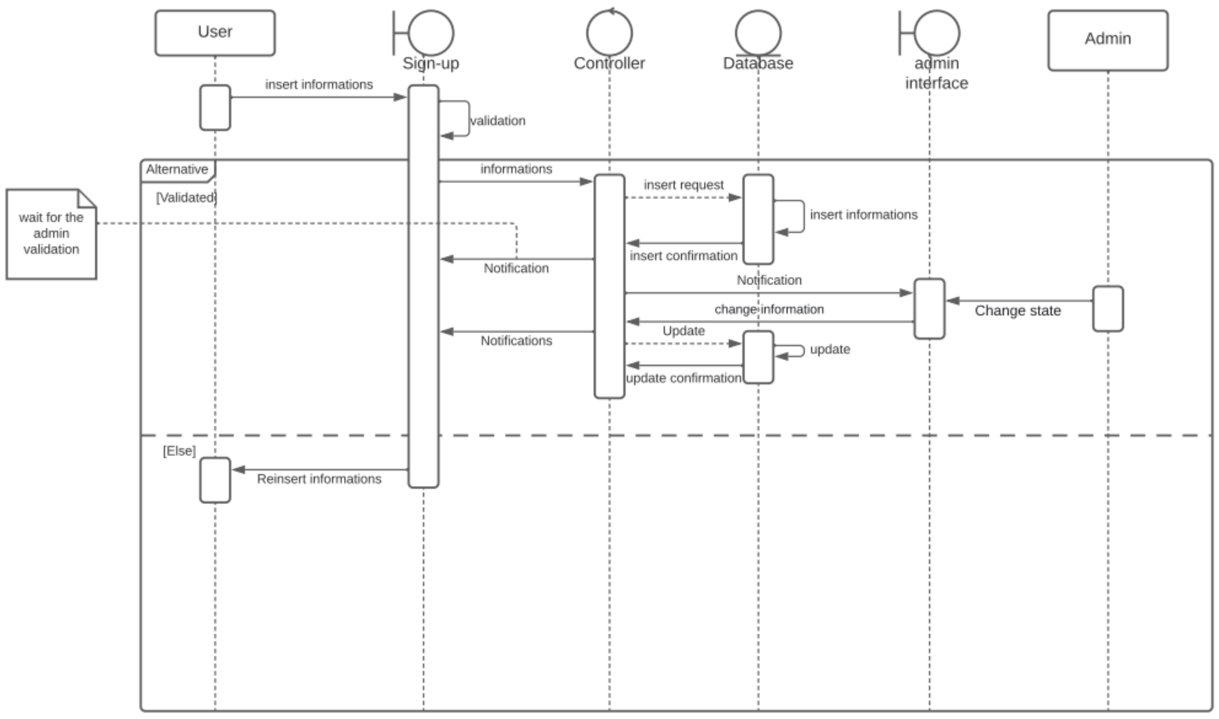


Figure 5 : Sign-up sequence Diagram

III.2.1.2.2 Description of the sign-in sequence diagram

Pre-condition: Sign-up

Main Actors:

- User

Secondary Actor: /

Scenario:

A user (a host, a client, an administrator, verification agent) must be signed-in to accomplish his tasks. So, to pass from the sign-in interface to the home interface, the inserted credentials must be validated.

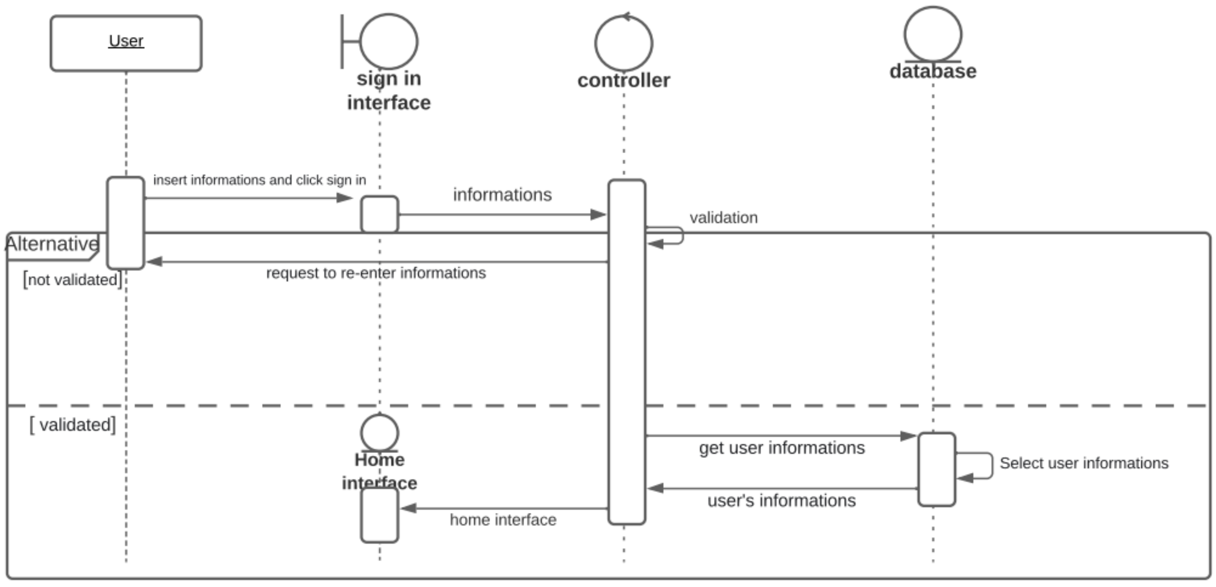


Figure 6: Sign-in sequence Diagram

III.2.1.2.3 Description of the add post sequence diagram

Pre-condition:

- Authentication of the host.

Main Actors:

- Host.

Secondary Actor:

- Administrator.

Scenario:

To publish a post or a location for rent in RENTED website, the host is asked to enter specific information about the post in the destined interface. A front verification will be first done. If the first entry is not accepted, the user is asked to re-insert the data until it is the correct one. When this step is finally validated, the database's fields can be filled successfully and a notification is sent as feedback to the host informing him what follows “The post is added and it is waiting for the admin’s confirmation to be published”.

For better security, the post is not directly published as said in the notification, it must be validated, verified and checked by the admin.

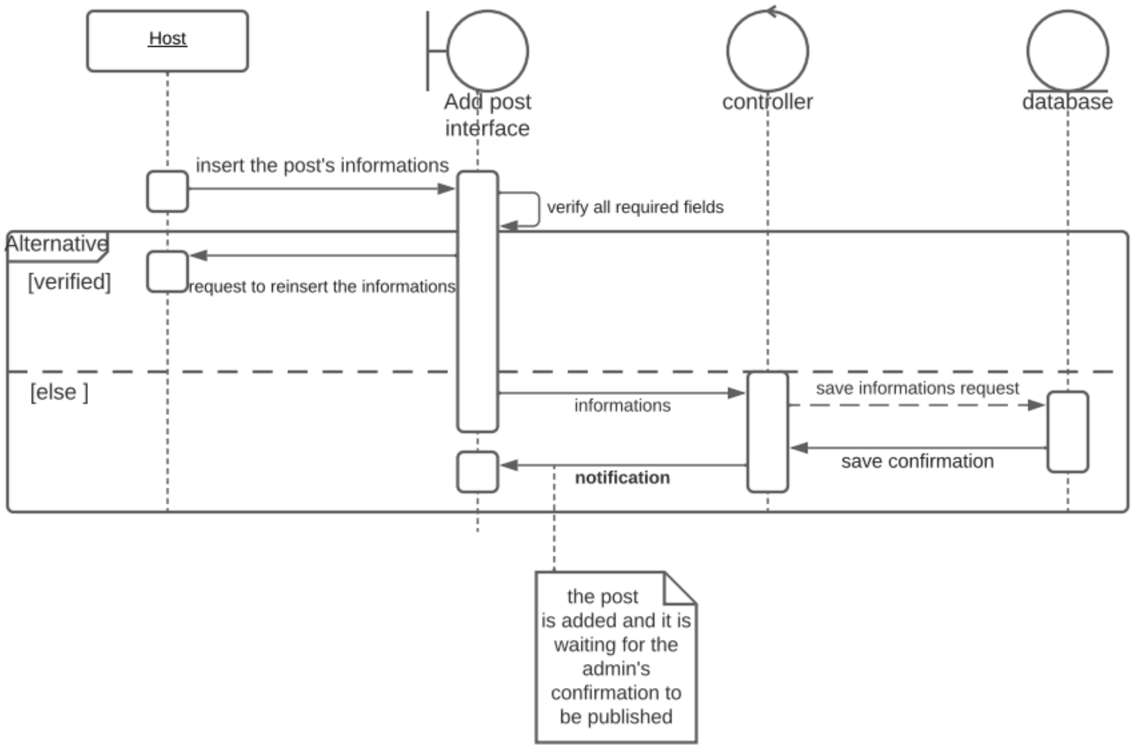

Host
Add post interface
controller
database
insert the post's informations
verify all required fields
Alternative
[verified]
request to reinsert the informations
[else]
informations
save informations request
save confirmation
notification
the post is added and it is waiting for the admin's confirmation to be published

Figure 7: Add post sequence Diagram

III.2.1.2.4 Description of the update post sequence diagram

Pre-condition:

- Authentication of the host.

Main Actors:

- Host.

Secondary Actor: /

Scenario:

A host can insert wrong information by mistake about a location, so being able to accomplish an update is essential, but it is only possible before the validation. After the edit, a notification would be sent to the host as feedback.

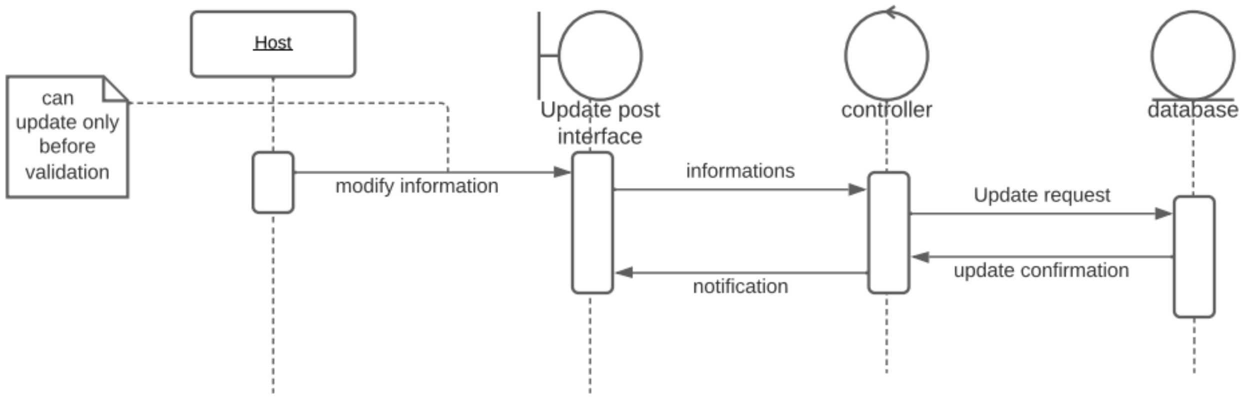


Figure 8: Edit post sequence Diagram

III.2.1.2.5 Description of the delete post sequence diagram

Pre-condition:

- Authentication of the host.

Main Actors:

- Host.

Secondary Actor: /

Scenario:

To delete a published location, some pre-verifications must be done first. A rented location cannot be deleted until it is released. A none reserved location needs a deletion confirmation to avoid any type of mistakes not done on purpose.

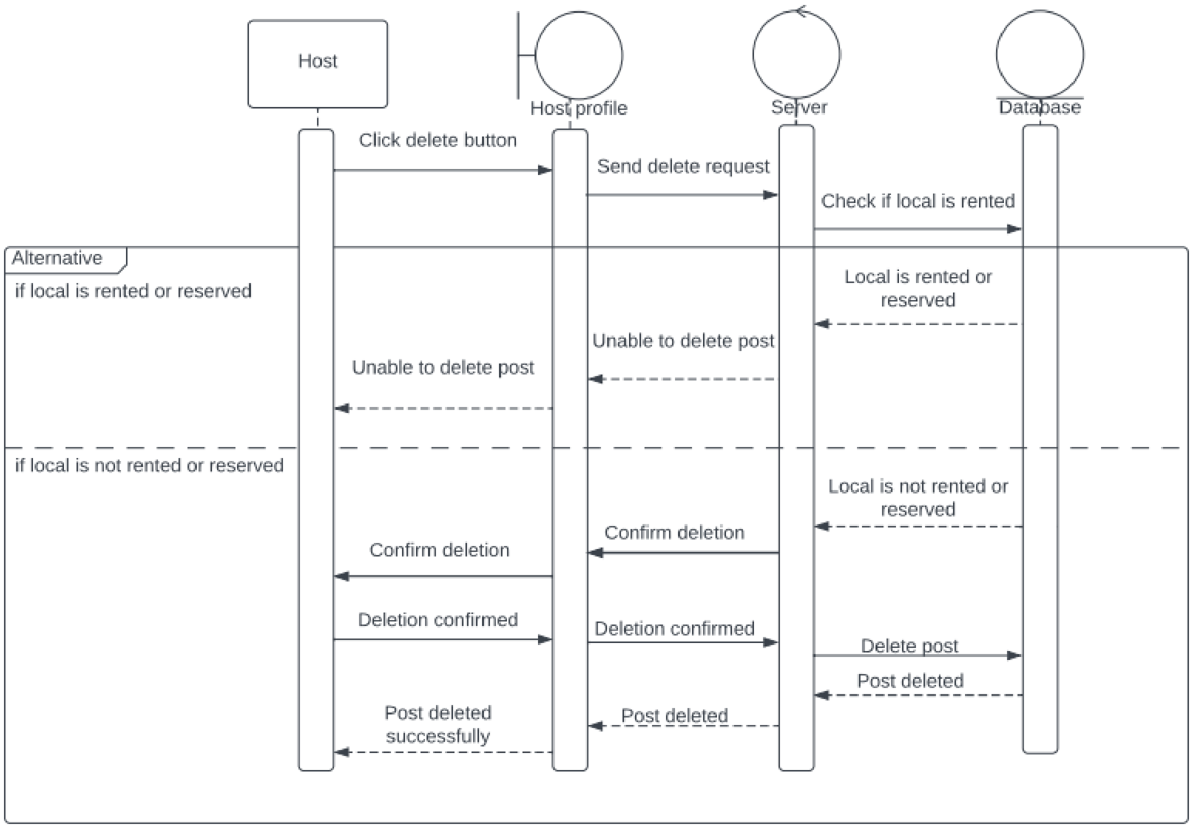


Figure 9: Delete post sequence Diagram

III.2.1.2.6 Description of the add reservation sequence diagram

Pre-condition:

- Authentication of the client.
- Posted location by a host.

Main Actors:

- Client.

Secondary Actor: /

Scenario:

Depending on the client's reservation information and preferences, a fetch in the database is done, if the characteristics entered are available in a local, the request would be pending until future validation, if not, a feedback informing the client that this type of local is not available for the moment would be displayed.

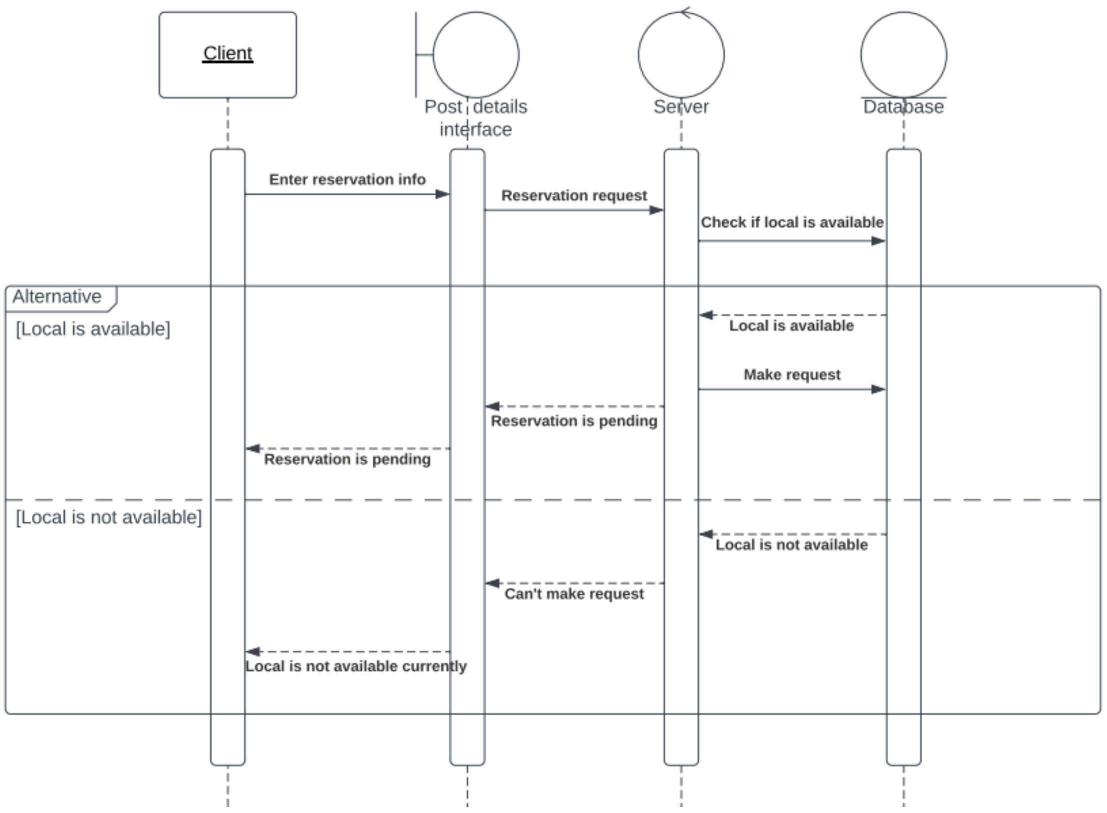


Figure 10: Add a reservation sequence Diagram

III.2.1.2.7 Description of the post verification sequence diagram

Pre-condition:

- Authentication of the admin.
- Authentication of the agent.

Main Actors:

- Administrator.
- Verification Agent.

Secondary Actor:

- Host

Scenario:

When a location is asked to be published, for better security and reliability, a verification, at a specific date, is done by an agent in real-life. He compares the images to the real post and see if it matches. The feedback is inserted in the database and sent to the administrator to edit the post status. If it has a positive feedback, the status would be edited to verified and the host would be notified.

And if it is declined, the reason of the decline is sent as well.

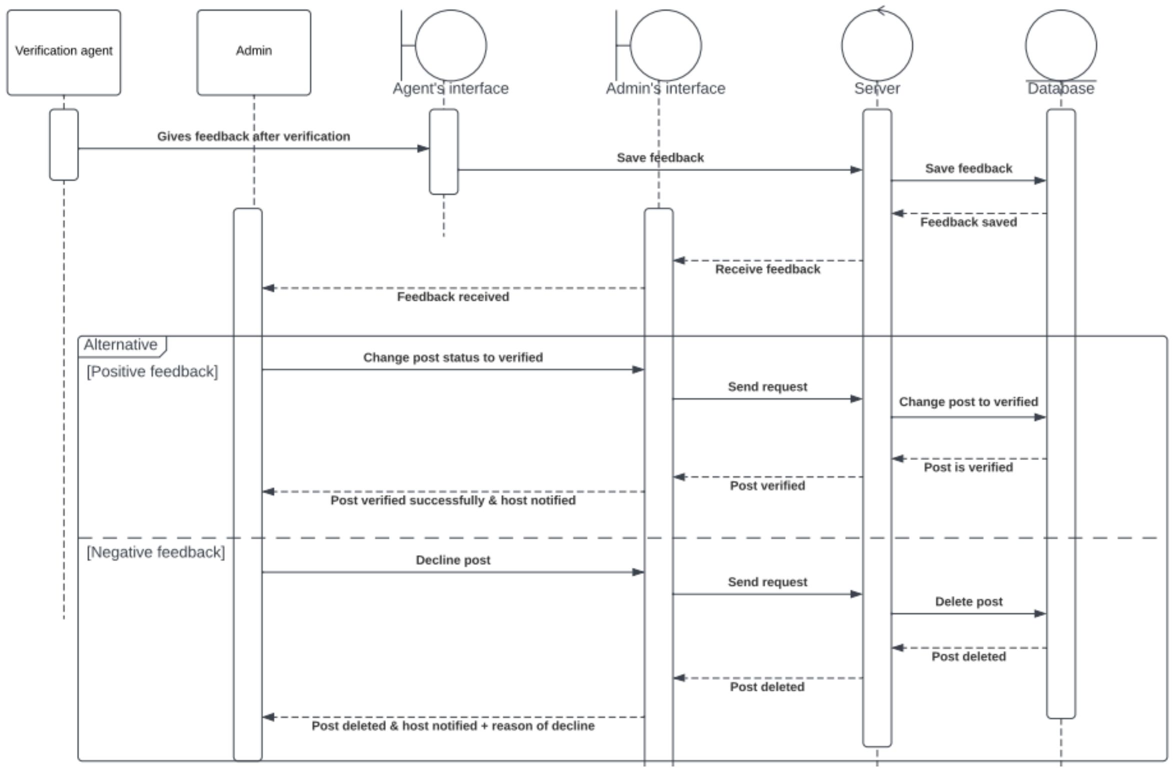


Figure 11: Post verification sequence Diagram

III.2.1.3 Class Diagram

A UML class diagram describes the object and information structures used in our website. It is generally recognized as the most important in the conception phase of a website. Its classes and relations can be implemented in many ways, such as database tables, XML nodes or software object compositions. In our case, each entity represents a database table.

III.2.1.3.1 Description of the class diagram

As shown below, there are a total of fourteen entities:

User, VerificationAgent, Host, Client, Admin, Complaint, RatingClient, Wishlist, Comment, Post, Images, RatingPost, Reservation, People.

We have four types of users, three inherit directly from the entity "User" which are: VerificationAgent, Host and Client. And from the "Client" entity, a Host entity is inherited.

- A host has the possibility to file a complaint as well as rating a specific client or not.
- A host can publish several posts. A post is published by a particular host.
- A client has the ability to add a post to his personal Wishlist and can also add a comment under that published post.
- A client main functionality is making one or many reservations and can additionally rate the concerned post.
- A reservation must include at least one person (entity people).
- An image is strongly related to a post. A post has from three to twelve images.

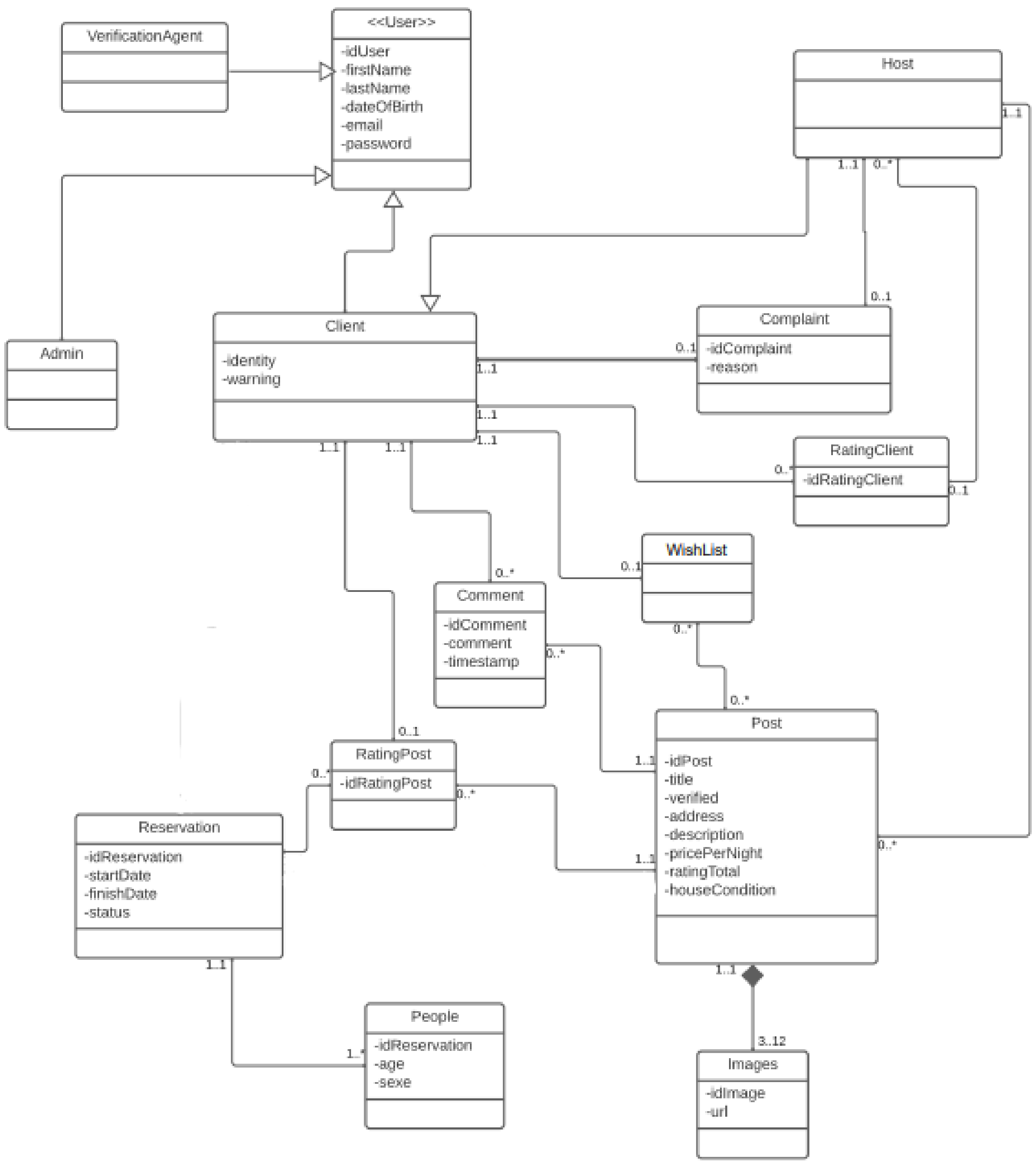


Figure 2: Class Diagram

III.3 Interfaces

III.3.1 The login interface

On the login page, the user fill in his address and password to be able to use our services. He can also reset the password by clicking on forgot password to receive a verification / reset email in case he forgot it.

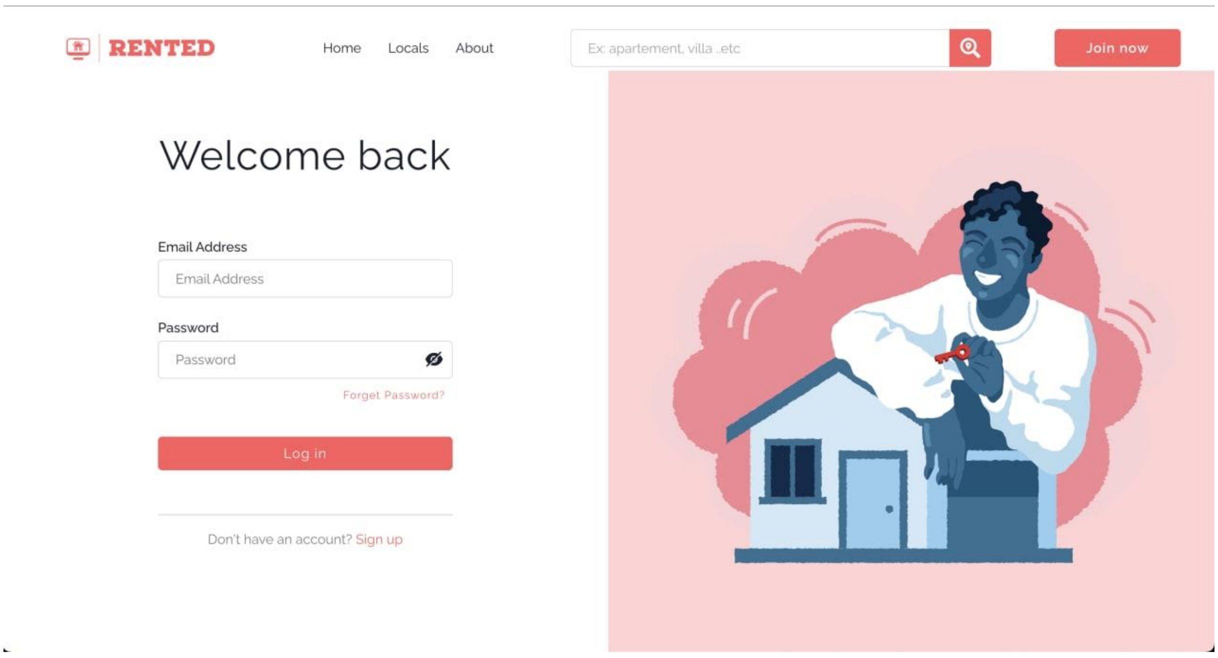
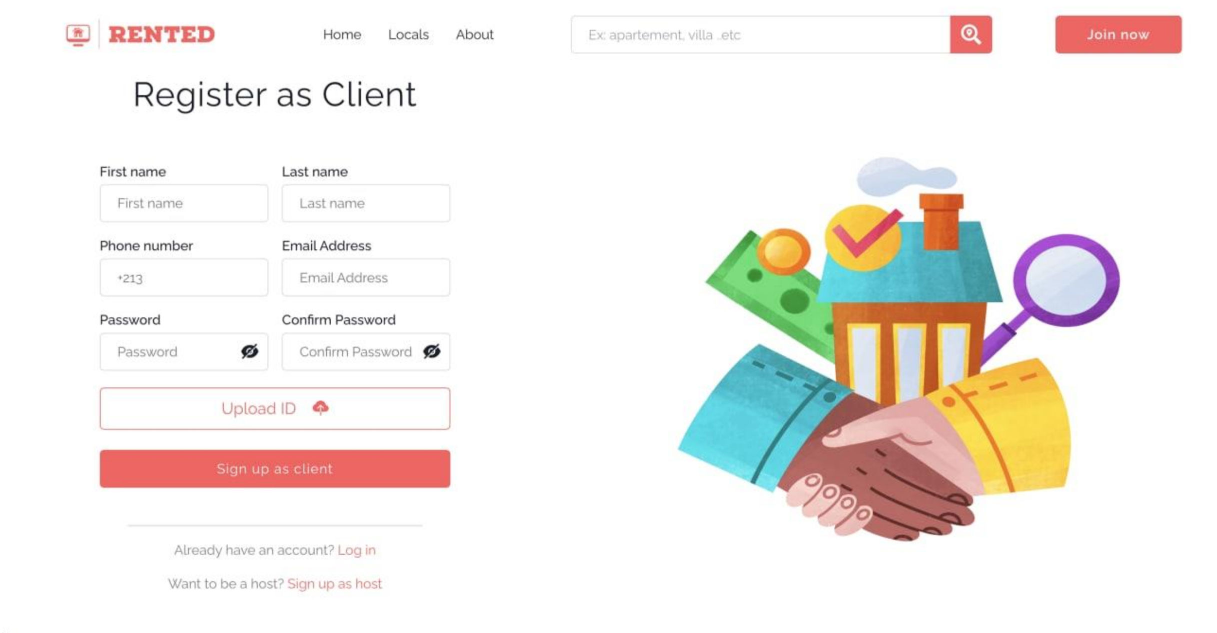


Figure 13:Login interface

III.3.2 The sign up interface

There are two versions of sign up interfaces (as host , as client) to don't confuse the client with the roles policy. However they have the same insertion process ; the user insert his credentials including his national identity , and a valid email address .When all the inserted values are valid an account is created and verification email is sended to the user's email address to step one verify his email , but when the admin check the identity , the account is fully verified and can use all its privileges.



Register as Client

First name: First name	Last name: Last name
Phone number: +213	Email Address: Email Address
Password: Password	Confirm Password: Confirm Password

Already have an account? Log in

Want to be a host? Sign up as host

Figure 14: Sign up as client interface

III.3.3 Landing page interface

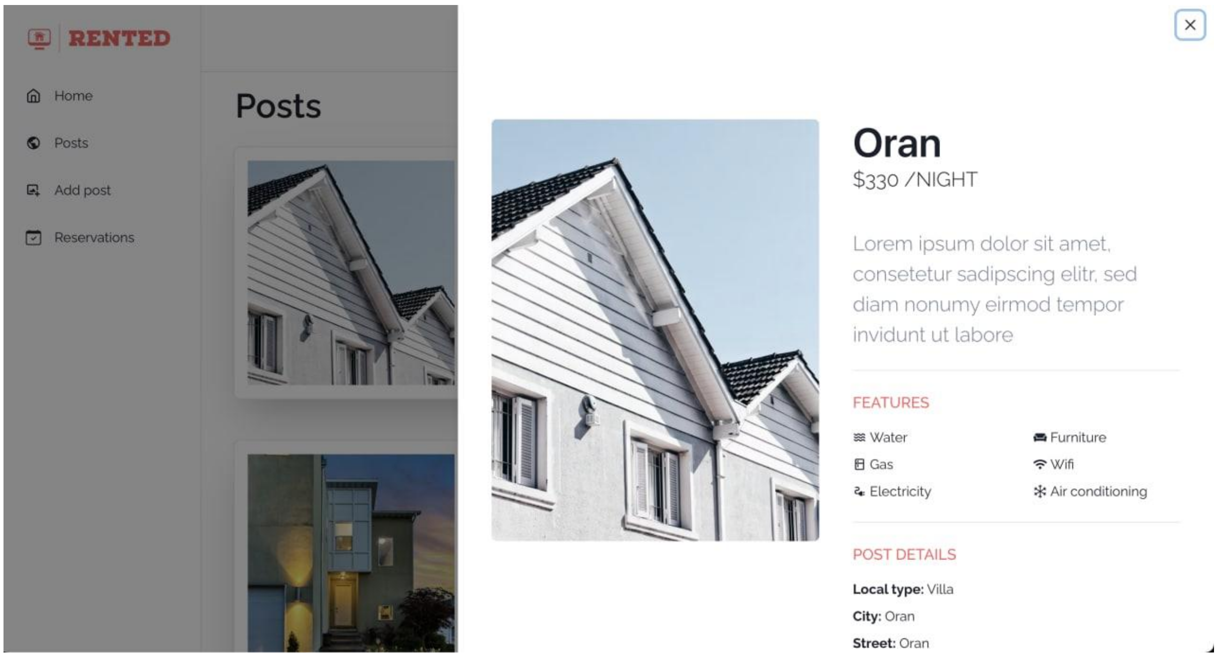
Landing page allow the user to see all the posts and he can use the filter or the search bar to make his experience much easier . Additionally , our website has a recommendation system based on what the user already liked or added in his wish list , so he can check that too in recommended for you section.

III.3.4 Host dashboard interface

The host dashboard provide him a group of functionalities to facilitate his interaction and increase his productivity in the website , so he can access to all his privileges in one single page.

III.3.5 Post interface

In post section the host is able to review all his posts generally /with details , and he is also allowed to make modifications in some fields if the post still not verified .



Oran

$330 /NIGHT

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore

FEATURES

Water	Furniture
Gas	Wifi
Electricity	Air conditioning

POST DETAILS

Local type: Villa
City: Oran
Street: Oran

Figure 15: Post interface

III.3.6 Add post interface

In this interface the host insert the post informations which based on them the agent gives his feedback .

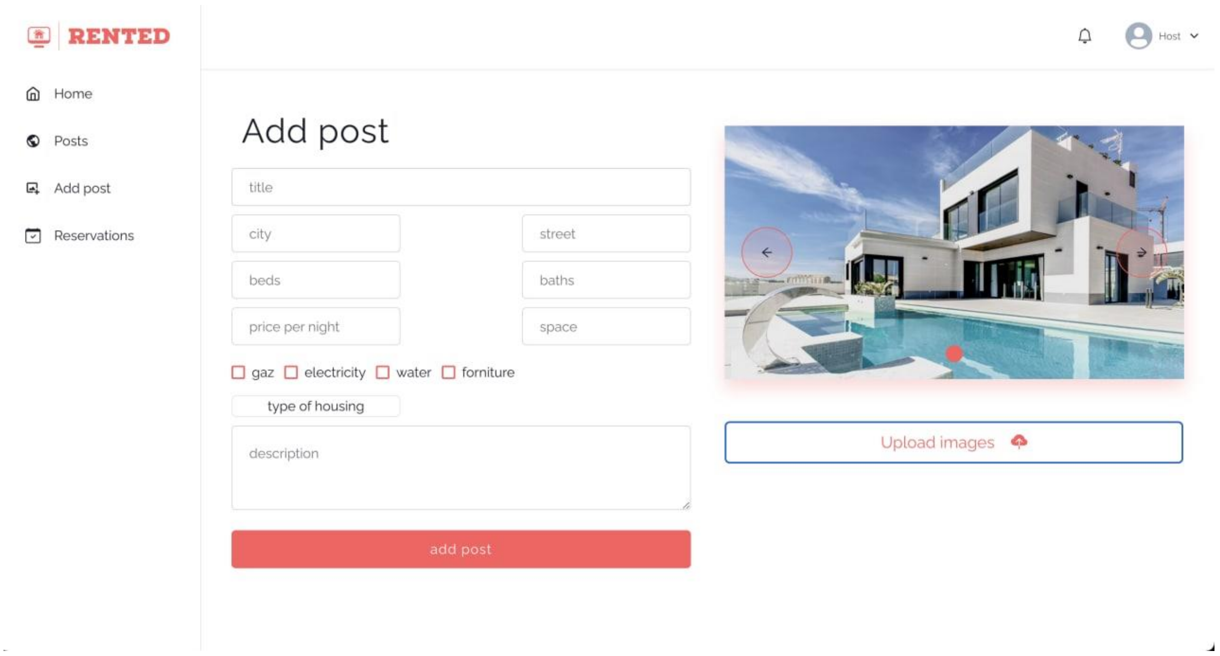


Figure 16: Add post interface

III.3.7 Reservations interface

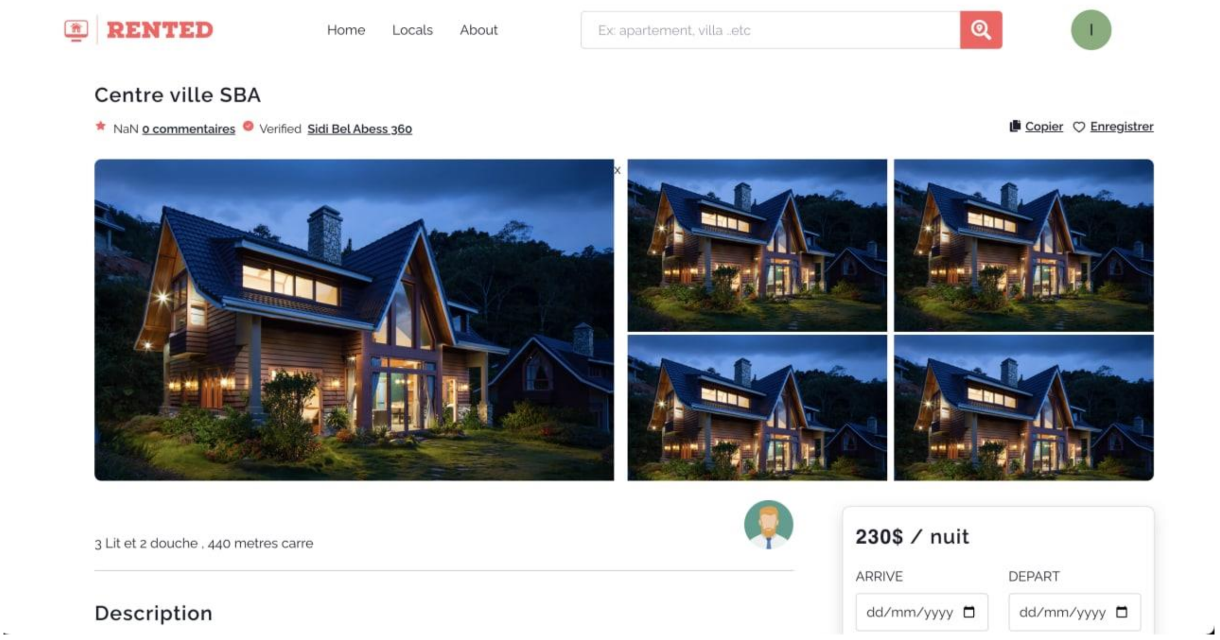
The host check all his posts reservations to see the current state of his locations and their rent history.

III.3.8 Client dashboard interface

When the client account is validated the website provide him a dashboard to use some services ; add reservation , track it and add comment,rate or even claim about a service.

III.3.9 Add reservation interface

If a client decides to book he just need to insert some informations about the people will spend time in the location and period he wants to reserve in .He can't choose a taken period as mentioned in the calendar.



Centre ville SBA

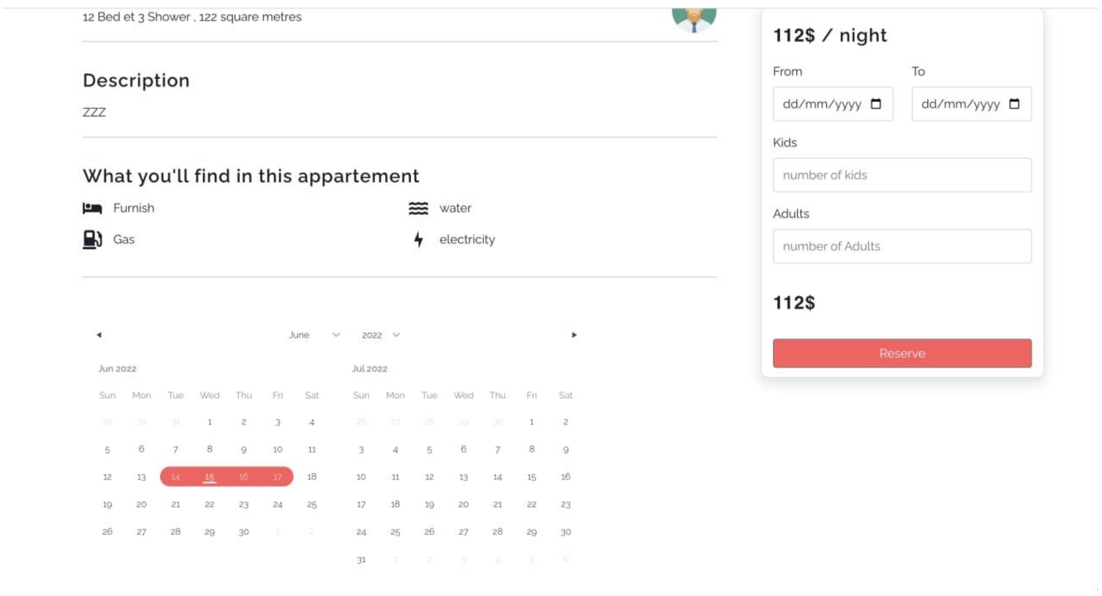
NaN 0 commentaires Verified Sidi Bel Abess 360

3 Lit et 2 douche , 440 metres carre

Description

230$ / nuit

Figure 17: Add reservation interface



12 Bed et 3 Shower . 122 square metres

Description

zzz

What you'll find in this appartement

Furnish
water
Gas
electricity

112$ / night

From: dd/mm/yyyy
To: dd/mm/yyyy

Kids: number of kids

Adults: number of Adults

112$

Figure 18: Add reservation details interface

III.4 Microservices

Microservices refer to both an architecture and a software development approach that consists of breaking down applications into the simplest elements, independent of each other. Unlike a traditional monolithic approach, where all the components form an inseparable entity, microservices work together to accomplish the same tasks, while being separated. Each of these components or processes is a microservice. Granular and lightweight

III.4.1 Why ?

After a long study of the subject and all its aspects, we found that the micro-services architecture is the suitable for this type of applications for the following reasons:

III.4.1.1 Improved Scalability

The capacity of each microservice to run autonomously makes it relatively easy to add, remove, update, and scale individual microservices.

III.4.1.2 Better Fault Isolation

With a microservices architecture, the failure of one service is less likely to negatively impact other parts of the application because each microservice runs autonomously from the others.

III.4.1.3 Technology Agnostic

When creating a microservices-based application, developers can connect microservices programmed in any language. They can also connect microservices running on any platform. This offers more flexibility to use the programming languages and technologies that best fit the needs of the project and the skillsets of your team.

III.4.2 Our microservices architecture

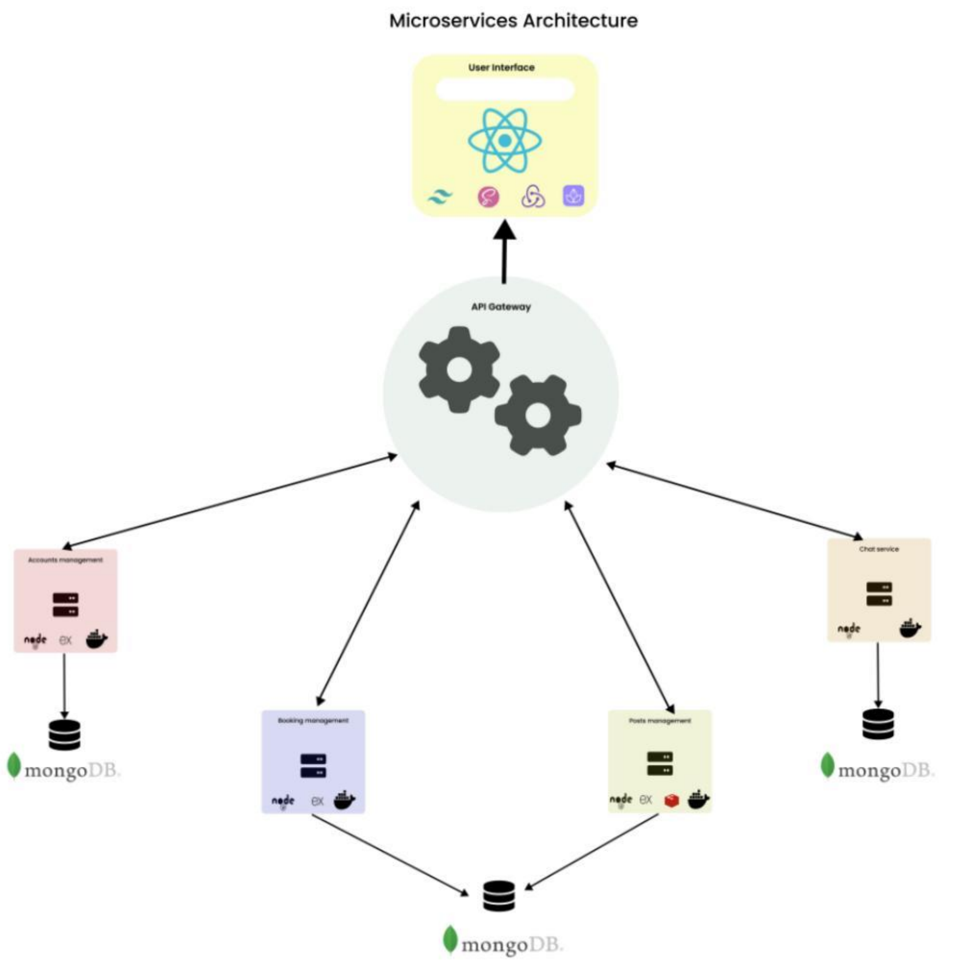


Figure 19: Micro-services diagram

III.5 Recommendation

III.5.1 Definition

Recommendation engines are a subclass of machine learning which generally deal with ranking or rating products / users. Loosely defined, a recommender system is a system which predicts ratings a user might give to aspecific item. These predictions will then be ranked and returned back to the user.

They're used by various large name companies like Google, Instagram, Spotify, Amazon, Reddit, Netflix etc. often to increase engagement with users and the platform. For example, Spotify would recommend songs similar to the ones you've repeatedly listened to or liked so that you can continue using their platform to listen to music. Amazon uses recommendations to suggest productsto various users based on the data they have collected for that user.

Recommender systems are often seen as a "black box", the model created by these large companies are not very easily interpretable. The results which are generated are often recommendations for the user for things that they need / want but are unaware that they need / want it until they've been recommended tothem.

III.5.2 Recommendation types

There are many different ways to build recommender systems, some use algorithmic and formulaic approaches like Page Rank while others use more modelling centric approaches like collaborative filtering, content based, link prediction, etc. All of these approaches can vary in complexity, but complexitydoes not translate to "good" performance. Often simple solutions and implementations yield the strongest results.

III.5.2.1 Collaborative Filtering Recommendation System

Collaborative filtering is the process of predicting the interests of a user by identifying preferences and information from many users. This is done by filtering data for information or patterns using techniques involving collaboration among multiple agents, data sources, etc. The underlying intuitionbehind collaborative filtering is that if users A and B have similar taste in a product, then A and B are likely to have similar taste in other products as well.

III.5.2.2 Content Based Systems

Content based systems generate recommendations based on the users preferences and profile. They try to match users to items which they've liked previously. The level of similarity between items is generally established basedon attributes of items liked by the user. Unlike most collaborative filtering models which leverage ratings between target user and other users, content based models focus on the ratings provided by the target user themselves. In essence, the content based approach leverages different sources of data to generate recommendations.

III.5.2.3 Hybrid Recommendation System

Various methods of recommendation systems have their own benefits and flaws. Often, many of these methods may seem restrictive when used in isolation, especially when multiple sources of data are available for the problem.Hybrid recommender systems are ones designed to use different available data sources to generate robust inferences.

III.5.3 RENTED recommendation system

Collaborative filtering based recommendation engine and NPM module builton top of Node.js and Redis. The engine uses the Jaccard coefficient to determine the similarity between users and k-nearest-neighbors to create recommendations.

PROJECT REALIZATION

IV Production and Realization of the website

In our web site the reservation is considered the main process and it is strictly controlled , well defined to ensure our confidentiality as well as the user's safety . It starts by authentication phase ; the user enter his credantials including his nationnel identety and valid email address .The web site automaticlly check his informations and if everything correctly mentioned a validation email will be sent to him . The account still not valid until the admin verifey the inserted identety manually for more security. If it's credible the account state change and now the user can benefit from the services under his role privileges .

There are four predefined user roles to split privileges and maintain traceability in the website:

IV.1.1 Admin:

The admin is the main character in the website. His mission is to keep things clear and control the process via a dashboard that allow him to check a user(client or host) identity, assign post verification to the nearby agent, validate the post to publish based on the agent's review, see all website stastics to make it easier to overview the progress. Also he is able to check the claims that users are making and send emails to both sides trying to solve the problem .

IV.1.2 Client

The client is the real user, he consumes the host's posts based on his needs by adding a reservation and the website gives him the ability to track his reservation statuses via the notification system.If he already booked into a post he can claim about bad host treatment, add a rating or commenting on his experience .

IV.1.3 Host

Host is the premium version of the client; When the client upgrades his account to the host role in addition to the client privileges, he will be able to add a post request and when the post is verified by the admin he can see the reservations on it , choose the appropriate request and reject others .Also in aim to make it easier, we provided him an interface so that he can review all his posts and reservations.

IV.1.4 Agent

The agent is the coordination between the host and the administrator.First he checks the posts non verified and then makes an appointment with the host to review his post.When the post is reviewed the agent send a feedback to the admin who's the responsible to validate it .We consider the agent’s role very sensitive cause based on his review the admin take decision to accept the post or not , that's why the admin should wisly choose the agents of the web site.

IV.2 The used technologies

IV.2.1 Front-end

- **React js**
- **SASS**
- **Redux**
- **chakra UI**
- **Tailwind css**

IV.2.2 Back-end

- **nodejs**
- **express**
- **mongodb/mongoose**
- **spring cloud gateway ,**
- **docker**
- **jwt (json web token)**
- **Redis**
- **serverFirebase**
- **Nodemailer**
- **raccon.js Engine**

IV.2.2.1 Raccoon.js

An easy-to-use collaborative filtering based recommendation engine and NPM module built on top of Node.js and Redis. The engine uses the Jaccard coefficient to determine the similarity between users and k-nearest-neighbors tocreate recommendations. This module is useful for anyone with users, a store ofproducts/movies/items, and the desire to give their users the ability to like/dislike and receive recommendations based on similar users. Raccoon takescare of all the recommendation and rating logic. It can be paired with any database as it does not keep track of any user/item information besides a uniqueID.

IV.3 The project progress

IV.3.1 Group organization

From the begging of the project there was an equitable division of tasks between team members , for a method of work we opted for agil (scrum) methodology to manage the project by breaking it up into several phases .

Once the works begin we started by specifying all the application requirements then we moved to the modilisation of all conception diagrams (sequence , use case , class) all the members together .When finishing this phase we moves to the most important phase witch id the development .In this step all the members contributed in both front-end and back-end development ,at the same time the team leader proposed courses for the team members to improve their skills .

IV.3.2 Personal Contributions

There is some words written by our team members to describe their experience :

IV.3.2.1 Ilyes Reda MERZOUG :

I definitely learned a lot throughout this project, first of all, managing a team is harder than it seems because it requires the project manager to know the capacity of each member as well as their strength for a better task division. secondly, the period of the project is approximately 3 months so throughout this period patience and consistency are required because the team's motivation is changing, so its the team leader's task to make sure to lift the spirit of work and motivate his teammates to keep the flow of work going. thirdly being the project's administrator at GitHub helped me a lot in learning the skill of reading code and understanding every piece of it at every part of the project meaning FrontEnd, Backend & DevOps.

finally, my advice for future team leaders is to simply understand the project very well and choose good members and then during the project never lose motivation ;).

IV.3.2.2 Dacine Yousra BOULMA :

As a member in this team i've learned so much from this project .

Beside technical skills i learned that soft skills are menditory to the integrity of the team , because working on team is more than submitting a work into a repository , it means replaying on several other skils like commination , time management so the whole team can progress .

In addition , i realized the importance of the role of the team leader and his efforts. To make a equitable division of tasks and explain exactly how the work should be donne .

IV.3.2.3 Mohemmed Zinnedine FEDDAG :

During this project , i faced a lot of real life problems ... handling each of them at it's own gain me much experience i did not had before . Plus working with the team was really a good boost to both technical and soft skills .

IV.3.2.4 Abir BEN AISSA :

This project was a unique experience that allowed me to learn a lot of things. On the one hand, I developed myself on the technical side and I learned new skills like the use of many frameworks for application development, node js for the back end and react for the front, I also got introduced to new concepts like microservices,gateway . . On the other hand I have developed my soft skills such as communication through group work, that wasn't a hard task to accomplish thanks to our team leader ,time and stress management. Finally, this project allowed me to have a better idea of my preferences in my field of study.

IV.4 Future perception

To continue to improve our application , RENTED development team proposed some features to add in the future :

IV.4.1 Kubernetes

Kubernetes is an open source container orchestration platform that automates many of the manual processes involved in deploying, managing, and scaling containerized applications.

IV.4.2 Chat micro-service

This micro-service is dedicated to manage the massengrer system between our users to facilitate the web site use and increase the userexperience .

IV.4.3 circuit breaker

Our application uses micro-services architecture , witch means that some micro-services synchronously invokes other micro-services , and with the existence ofthe possibility that a micro-service is unavailable , so to provide a network or service failure from cascading to other services we should prevent a circuit breaker to Handel such case

IV.4.4 The load balancer

To optimizes the use of application delivery resources and prevents server overloads. Load balancers conduct continuous health checks on servers to ensure they can handle requests.

V General Conclusion

Rented is the perfect solution for people who want to rent their homes . Transactions are safe and made with confidence .

The apply of the internet-based technologies improves easy and rapid access to Rented services .

Rented has a lot of perspective to improve the quality of its service and to extend and enlarge the site, such as adding an internal chat service and an automatic authentication system.