PDG - Group 13

Students :	Tegest BOGALE
	Guillaume COURBAT
	Farouk FERCHIHI
	Alice GRUNDER
Professor :	Bertil CHAPUIS
Assistant :	Gaétan ZWICK

Table of content :

1. Description of the project

Problematic

Many pet owners encounter challenges when seeking appropriate pet sitters for their animals, particularly during travel or busy work periods. Traditional methods such as relying on friends or family, often prove limited and time-consuming. Moreover, a lack of transparency regarding potential pet sitter's experience adds to these challenges.

Solution

Presenting "PetKeeper", an android mobile app designed to effectively address these challenges. This app helps pet owners to easily find well-suited animal keepers to look after their pets when needed. The app also helps people who are willing to provide their service as a pet keepers to connect with pet owners. The app achieves transparency by allowing pet keepers to show their approved experience and qualifications.

Functional requirements

User registration and profiles

Users can register on the app without specifying their role (pet sitter or pet owner). During the registration process, users are required to provide and consistently update the following essential profile information:

- first name
- last name
- date of birth : used for age verification purpose
- gender: to ensure a personalised experience
- home address
- email address: used for communication and login
- password : user-chosen secure password for authentication

User login

Upon successful registration, users can effortlessly access their accounts using their registered email address and chosen password.

Role selection

Following successful login, users gain the ability to define their roles as either pet sitters or pet owners, aligning the app experience with their preferences.

Creating announcement

Pet owners can create announcements by specifying the pet type, location, date ,time and hourly price.
The following informations should be included for every animal:

- Name
- Type
- Race
- food habits
- vaccination
- photo
- other (for specific notes)

Search and matching:

Pet sitters can search for available announcements based on criteria such as location , availability, price and type of pet.
Searching results provide comprehensive information including owner name, pet type, location, price, rating and reviews.

Booking and scheduling

The app should enable pet sitters to get in contact with owners, by message

Messaging

An integrated messaging system allows pet owners and pet sitters to communicate and discuss details before finalising booking.

Rating and reviews

To establish a reputation system within the app, users can leave reviews and ratings for each other based on their experiences. This feature is activated only after a successful collaboration.
The evaluation has different criterias. For each criteria, it contains a grade between 0 and 5. The criterias are different for the pet owners and pet keepers. For pet owners it includes punctuality, loyalty and reliability whereas for the pet sitters it adds their engagement. Users may rate once and, if necessary, amend their evaluation.

Profile rating display

User profiles show an average of the various evaluation criteria grades.

Notifications

Users should receive notifications for booking requests, accepted bookings, messages and other relevant activities.

Killer feature

- Use AI to identify animals species on images
- (Real-time monitoring : GPS monitoring feature to allow pet owners to know the real-time location of their pets.)

Non-functional requirements

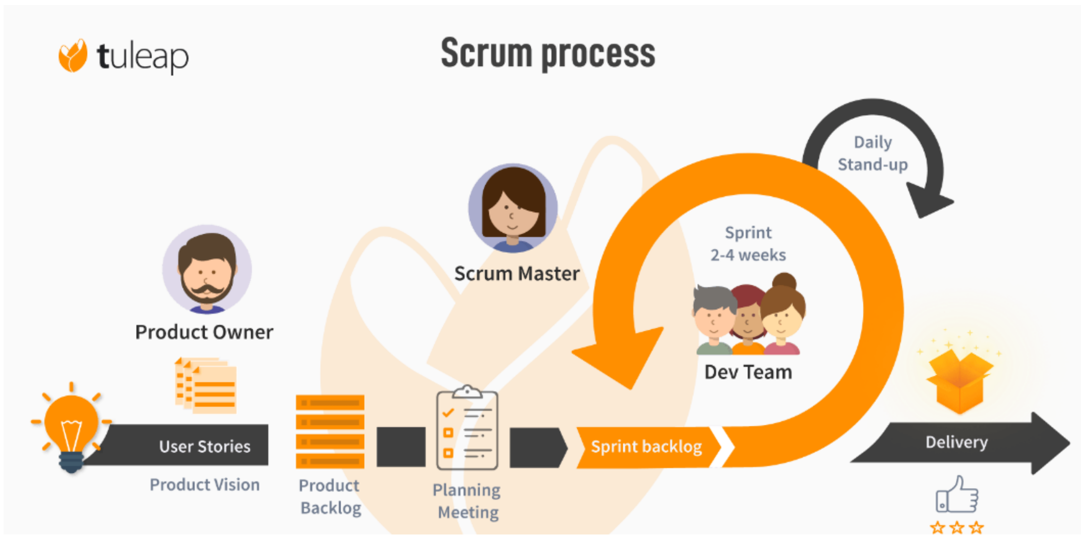
1. Security
    - **data encryption**: sensitive user data such as password should be encrypted both during storage and transmission. Personal information should not be visible for others and all the data should stay in europe.
    - **authentication and authorization** : ensure secure user authentication and proper authorization mechanisms
2. Performance
    - **Scalability** : the system should handle an increasing number of users and requests without degrading in performance
    - **load balancing**: incoming traffic should be evenly distributed across multiple servers to prevent overload
    - **response time** : the app should respond to user actions within a reasonable time frame and should be available 24/7.
3. Maintainability
    - **modularity** : the system should be designed in a modular way to be able to update individual components without affecting the entire system
    - **code quality** : employ best practices and coding standards to have clean and maintainable code.
    - **documentation**: maintain comprehensive documentation for the app's architecture, API's and configurations
4. Simplicity
    - The app should be easy to use and have simple interface design with intuitive navigation

2. Description of the development methodology :

In order to remain dynamic and participative in the conduct and management of the project, we have chosen to use the Agile scrum method.

Scrum:

The diagram below shows us the steps as well as the main actors of the scrum method :



picture source : https://www.tuleap.org/agile/agile-scrum-in-10-minutes

Step 1: The Product Backlog

In this phase, the Product Owner or in other words the customer (on the other hand it is us) expresses these needs. It identifies all product features through user stories in the Product Backlog (a kind of purchase order).

Step 2: The sprint

The SCRUM method is characterised by a distribution of each of the tasks to be done. We are going to sort the functionalities and tasks that it distributes in Sprints following that we organise the sprint planning which is a kind of negotiation between the product owner and the production team, the sprint planning meeting allows us to select in the product backlog highest priority requirements for the customer.

The Daily SCRUM:

In order to respect the SCRUM method, every morning, when the team is complete, we carry out the daily scrum: a meeting of 5–10 minutes, which is done standing up (to go faster!) where we talk of three things:

What did we do yesterday?
What problems did we encounter?
What are we going to do today?

Scrum board :

At the beginning of the sprint, we identify each of the small tasks that will be necessary to achieve the functionality in question of the product. Each task is represented by a coloured post-it. They are in the first column: to do.

As they are completed, they will be moved to the right. The team established how long each task would take. It is thus easy to see during a sprint if the team will stick to its timings and these objectives. This forces you to strive for efficiency more than perfection.

Story
To Do
In Progress
To Verify
Done
Scrum Board
TECHNO-PM

Step 3: Sprint Review

Every two days that closes a sprint we test the benefits of the feature with the Product Owner.

We make a demo of what has been created. Then the recipient of the functionality (the customer for example) confirms or not whether the functionality works as desired. The circle is complete !

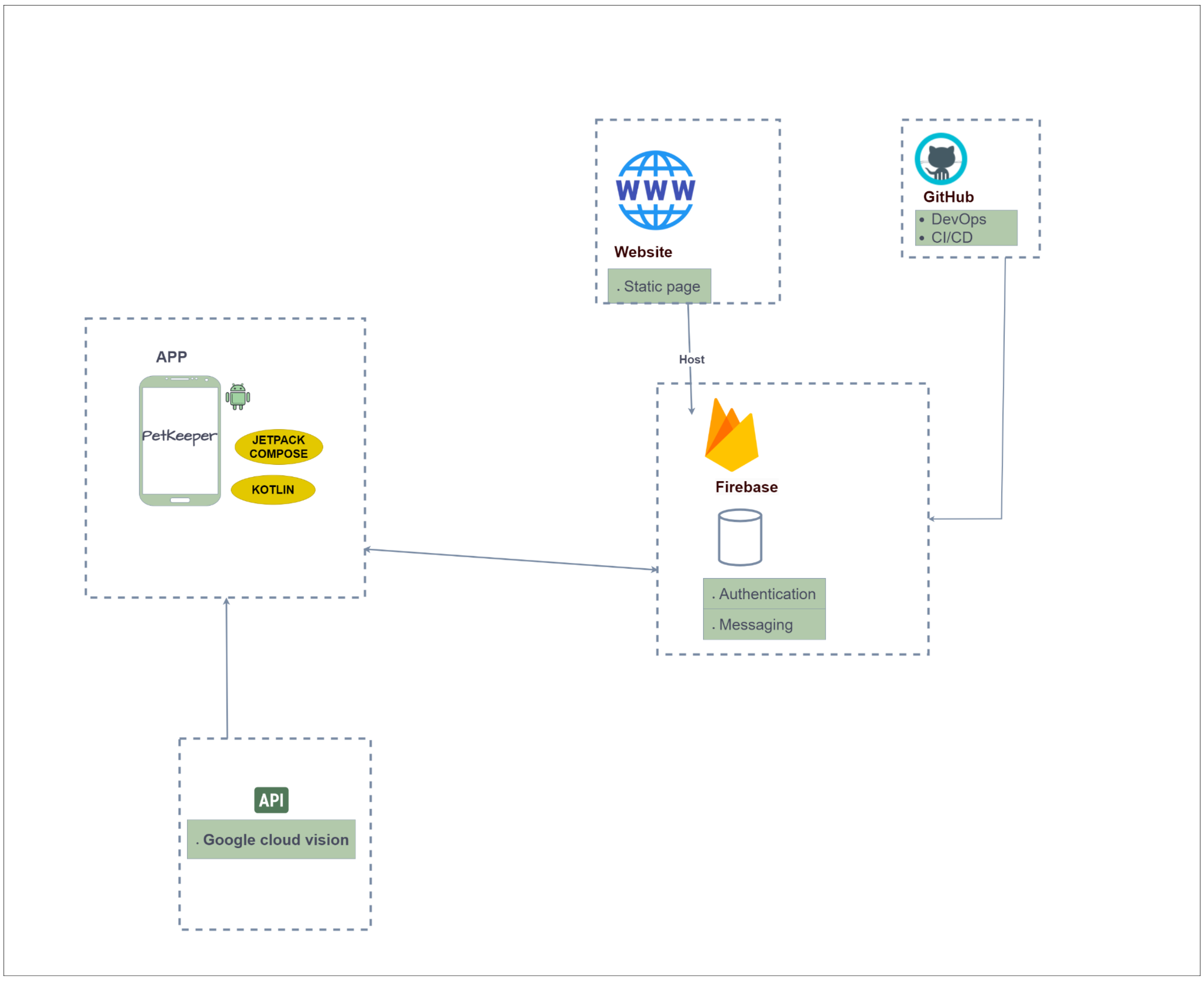
3. Mockups

Mockups are subject to change and won't necessarily represent the exact look of our app.
We use Figma. This is the link to our Figma project.
We chose to work with figma because it offers a lot of design possibilities and the handling is very easy

4. Landing page

This is the link to our landing page

5. Description of the architecture



6. Description of the technical choices:

- **Platform**
  We chose Android because it has a large user base and offers flexibility for app development.
- **Programming Language**
  We chose Kotlin because it is officially supported for Android app development and its syntax is more concise compared to Java.
- **IDE**
  We chose IntelliJ and Android studio because they are well-suited for kotlin development.
- **User Interface(UI)**
  We chose Jetpack Compose because it is a modern UI toolkit for building native Android interfaces using declarative syntax.
- **Database**
  We chose Firebase (Google Firestore), a NoSQL database
- **Authentication and Security**
  We chose Firebase Authentication & encryption because it helps us manage user authentication without having to build the system from scratch. We use gitflow to manage our branching on git. You can find the details here

7. Description of the work process:

We will use GitHub as our VCS (Version Control Software), it also has issue tracking as well as “projects” where we can do our scrum board integrated with our GitHub issues.
We will use a “git flow” practice and use a “devops” mind (CI/CD, agile development - scrum).

https://www.gitkraken.com/learn/git/git-flow

Meaning we have a “main” branch on which only functional, tested, and linted code goes.
From “main” we merge onto “Develop”; this is the branch on which we develop our app.
From this branch we make “feature” branches with a naming convention : “feature/[issue number]-[feature name]”
Once a feature is done it is merged into develop and when we want to release a new version of our app we create a “release/v[version number](-[alpha/beta])” where CI is executed and we deal with last minute linting and bugs. Then it is merged into main. We then tag this merge and our CD creates a release.