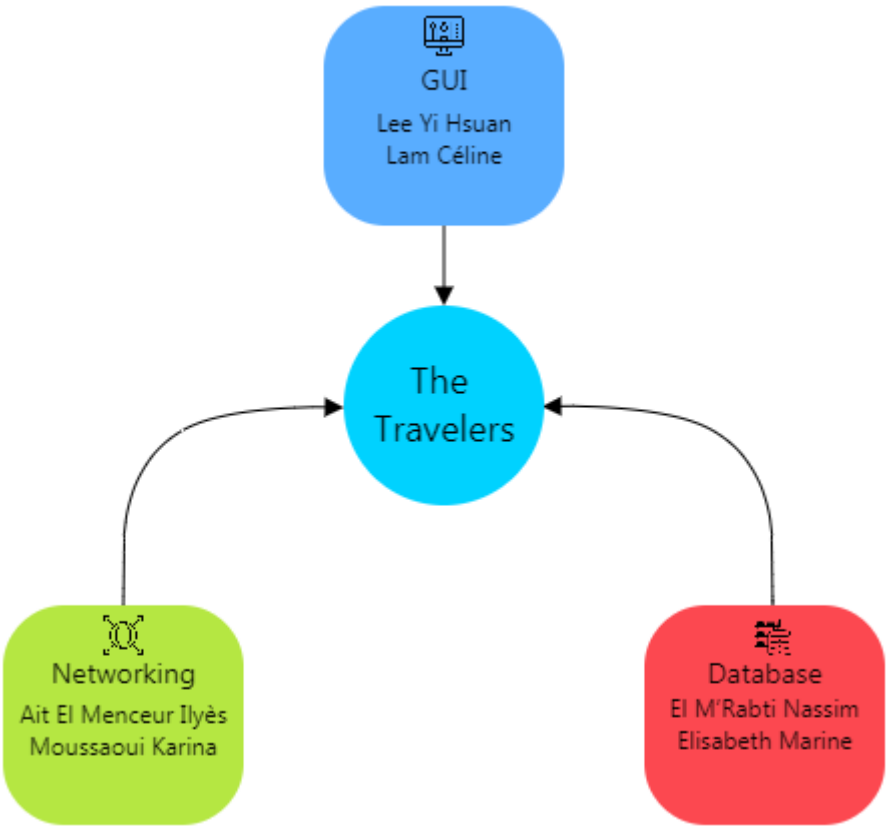
Ait El Menceur Ilyès
El M'Rabti Nassim
Moussaoui Karina
Elisabeth Marine
Lee Yi Hsuan
Lam Céline

Software engineering project

https://github.com/AitElMenceur/The-Travelers.git

The Team



Ait El Menceur Ilyès
El M'Rabti Nassim
Moussaoui Karina
Elisabeth Marine
Lee Yi Hsuan
Lam Céline

The development method

We used Kanban as a development tool because it is easier to review the process and make improvements that cut waste, streamline workflow, and reduce overhead. It also empowers teams to make more agile decisions.

In the database team, we did a practice of XP extreme programming (1 person review and the other code). Gitflow was also used, we had branches like networking, GUI, and Meetings were done using Teams. We split the workload into 3 teams: Networking, Database and GUI.

Architectural style and design choices

We used a 3- Tier Layered Architecture (Database->Networking->GUI). This architecture was interesting in this case because it enabled us to have 3 functional and independent Parts for this project, so in terms of project management, it was convenient to decompose the project. The architecture also allows us to use multiple Hardware for each layered, for instance for a larger deployment, server farm, and Storage unit would easily implement our software.

Also, the Networking team of the project decided to use a Client-Server Architecture. We chose this architecture because we wanted to improve CPU performance for the client which might use different kinds of hardware: smartphone, computer...

SOLID principles

We tried to respect single responsibility principle by creating different classes, for instance, CommunicationHandler and Portlistener were separated from the server, we did consider that the client goal was to send request, so we did not separated request method from the Client class as it wouldn't fit in this definition.

The design is Open for extension, close for modification: New type of data can be added but the old one is protected, also new servers can be easily implemented thanks to abstractions

We tried to keep interfaces as "slim" as it could get, for instance, we limited interfaces to critical methods and for testing

We tried to Design the object that would be sent over the network by following the Liskov principle: here every Data child class can be cast as Data Without any issue

Concerning the database, we tried to respect the singleton principle, so it guaranties us to have only one XML builder, which will optimize the performances of our program and will put a lighter charge on the CPU.

The single responsibility principle is also respected. Indeed, our methods only do what they are supposed to do.

Also, our program is opened for extension, we can implement some new date without "breaking" it, and it is also closed for modification. Except our Xml, the other type of data is protected.

IDE project

Team members used their favorite IDE: Eclipse and Vscode. Both allow an implementation of maven,github and javaDoc generator.

Ait El Menceur Ilyès
El M’Rabti Nassim
Moussaoui Karina
Elisabeth Marine
Lee Yi Hsuan
Lam Céline

Design pattern

Singleton for clients in ordrr to have a single port used by the client and prevent memory leaks . We wanted to add this design to the server , however it is a multi port server so the design wasn’t possible here.

Singleton for database.

Ait El Menceur Ilyès
El M'Rabti Nassim
Moussaoui Karina
Elisabeth Marine
Lee Yi Hsuan
Lam Céline

	Céline	Ilyès	Karina	Marine	Nassim	Rebecca
Rolls	GUI team	Networking Team	Networking Team	Database Team	Database Team	GUI team
Description of the role	GUI design	Network implementation	Network implementation	Database implementation	Database implementation	GUI design and implementation
Responsibilities	Discuss the correctness of data flow(GUI)	Development of the server and client Design a Solid Networking	UML documentation Code Reviewing	Implementation of database and code reviewing	1.Implementation of the code 2. Code Reviewing	1. Communicate with Network team and Database team. 2. Debugging
Tasks done	1.UML documentation 2. User and system requirement document	1.Development of the server and client. 2.UML class diagram with the team 3.Merging with Database and GUI 4. deploy maven project (packaging, testing, compiling) 5. Review documentation	1.Code reviewing Bug fixing on Networking. 2.UML documentation 3. Bug fixing 4.Review documentation	1. Code implementation. 2. Fixing bugs in the code 3. Review Nassim's code 5.Documentation 4. Sequence diagram UML 6.Review documentation	1. Code implementation 2. Fixing bugs in the code 3.Review Marine's code 4. Review documentation 5.Adding methods needed by the networking team	1. GUI 2. User and system requirement document