{EPITECH}

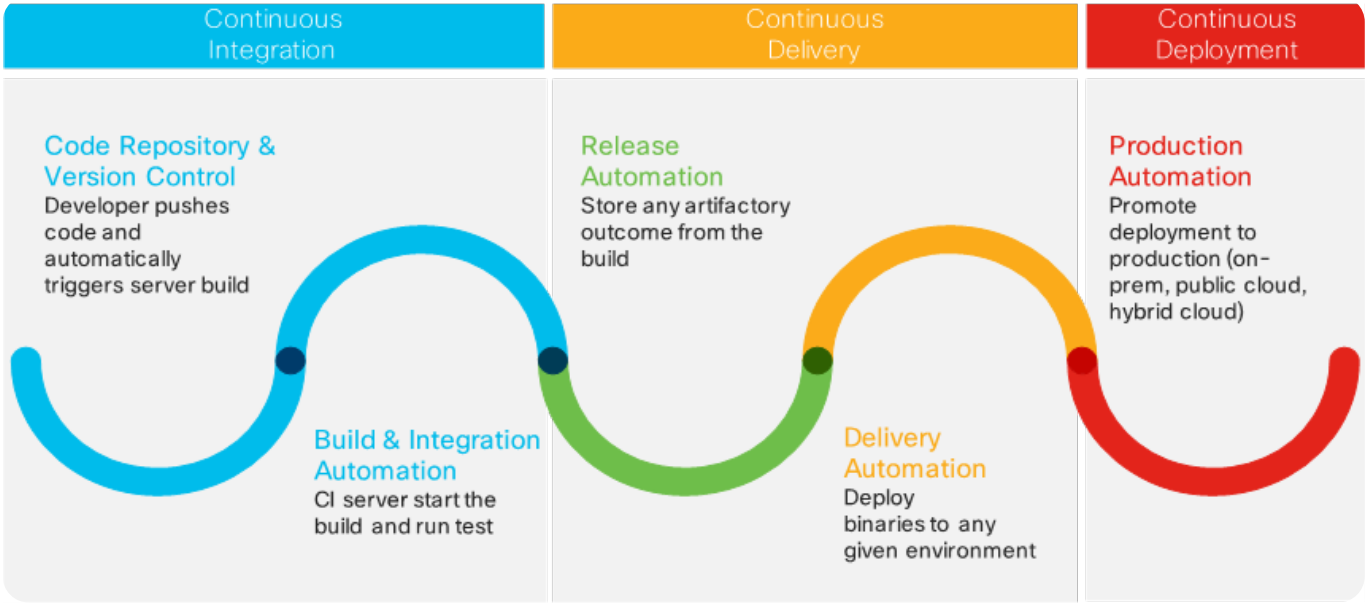
TIME MANAGER_

< TRINITY />

TIME MANAGER

Following the Dev OPS philosophy

Since you are the sharpest tool in the box, you take the excellent initiative of implementing the DevOps methodology for this new project.



Doing so, you will be more agile, reduce your delivery delays through automations that accelerate development and ensure the quality of your project.

Before starting to develop your software, it's important to start considering **which technologies to use and how to structure the project**.

The authorized technologies are listed in the subject for the backend and frontend parts.

Once you defined them, next step is to implement some good DevOps practices. Set up your environments is crucial as well as define how you will collaborate and what are the tasks that you will do.

We all want to go fast in the action of the the code, but more time you spend on planning, less time you waste on unpredictable problem.

Project Construction

As part of this project, you will have to develop two independent parts. The authorized technologies are listed in the subject for the backend and frontend parts.

To homogenize the project's construction, you will use Docker and Docker Compose.

You must dockerize your backend and your frontend, if possible.

If you choose to do a mobile client your docker image must handle the build of your APK.

Build your project

You will have to make a compose.yml file at the root of your project, which will describe the different docker services used.

This file must include at least the following docker services:

- ✓ a **backend** service to launch the **backend server**;
- ✓ a **frontend** service to launch or build the **frontend**;
- ✓ a **databases** service to launch the database;
- ✓ a **reverse proxy** service to:
  - allow access to the backend and frontend services;
  - expose a public port.

Check the documentation about **network**, **depends_on**, and **multistage**.

Your project is intended to be able to go into production, prepare the ground by creating a second Docker Compose file for this environment.

Build for production implies several aspects to take in consideration.

docker-compose up

Validation of the integrity of your images will be done when launching the **docker-compose up** command.

The following points must be respected :

- ✓ the **backend** service will run and exposed on the port of your choice;
- ✓ the **frontend** service will run and exposed on the port of your choice;
- ✓ if the client is a **mobile** application , it must build and provide the version of mobile client service;
- ✓ the **database** service will must persist the data.

Testing

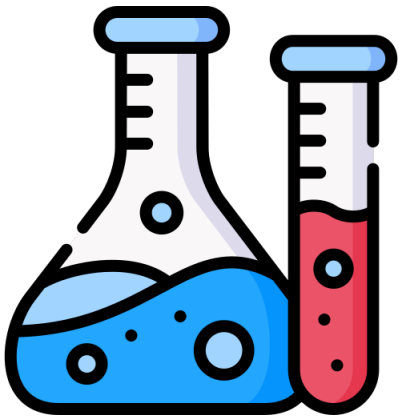
To ensure the quality of a project, testing it is important. You can do it manually but it will be quickly time consuming...

To assert the behaviour of your backend application and avoid regression you're expected to write tests. Your backend which is the core of your application containing the business logic must be tested.

You're expected to test as many routes as possible of your API.

Untested routes must be justified.

For each framework, there are testing tools.



Pipeline

You will work with GitHub (organization EPITEHCMSC) as your source control management platform.

To facilitate collaboration it's important to define your git flow and configure your repository in consideration: branch protection, commit message norm, merge request configuration, ...

How to contribute to your project could be a nice information to have in your README for other developers.

Working on your project, adding new features and ensuring the quality needs several repetitive manual tasks: building, launching tests, etc.

The automation must be your new best friend, saving time and avoiding human errors.

Using **GitHub actions**, you **MUST** create a pipeline with the following jobs:

- ✓ build your frontend and backend;
- ✓ launch your tests;
- ✓ create a test coverage report.

The logs of your CI must be managed carefully.

What is a mistake if everything is a mistake ?

GitHub and its environment are very powerful, and the possibilities are plentiful. Don't hesitate to explore them.

Backend application

Your application will allow your employees to put their arrivals and departures and their manager to handle the teams and see KPIs of company members.

Manage employees in teams and handle schedules need a lot of logic and need to be develop carefully avoiding errors in work hours count that can lead to strikes.

Mainly, your application allows the employees of a company to:

- ✓ report their arrival and departure times;
- ✓ view a summary of their working hours.

As said earlier your application will be composed of two main parts:

- ✓ the backend application;
- ✓ you frontend client.

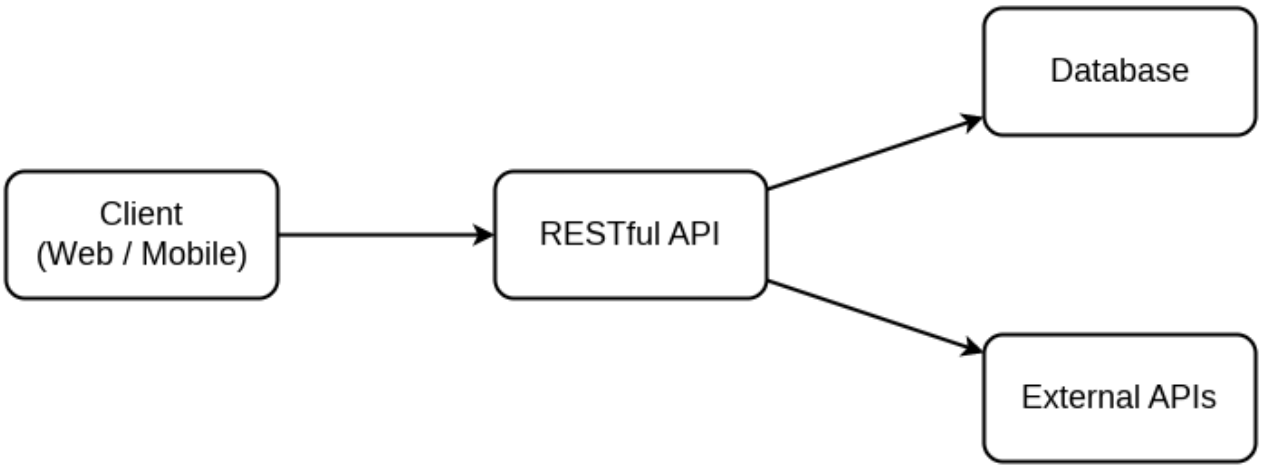
All the business logic and features of your application will be contained in your backend server.

The backend is the core of your application.

The backend must be exclusively an API.

API

Your **A**pplication **P**rogramming **I**nterface must be RESTful.



The user's (employees and managers) content allows recording users' information, such as:

- ✓ first name;
- ✓ last name;
- ✓ email;
- ✓ phone number.

The team's content allows recording some information about a team, such as:

- ✓ name;
- ✓ description;
- ✓ members;
- ✓ manager.

You must implement, at least, the following routes and endpoints:

- ✓ `GET /users` retrieves the users;
- ✓ `POST /users` adds a user;
- ✓ `PUT /users/{id}` updates a user;
- ✓ `DELETE /users/{id}` deletes a user;
- ✓ `GET /teams` retrieves the teams;
- ✓ `POST /teams` adds a team;
- ✓ `PUT /teams/{id}` updates a team;
- ✓ `DELETE /teams/{id}` deletes a team;
- ✓ `POST /clocks` Set the arrival / departure of the authenticated use;
- ✓ `GET /users/{id}/clocks` Get a summary of the arrivals and departures of an employee;
- ✓ `GET /reports` Get a global report based on the chosen KPIs

Roles

Your application is aimed at two kind of users: **employees** and **managers**.

These two roles will allow users to have access to specific features.

Below is a list of features that you should implement:

- ✓ Common features:
  - edit their account informations;
  - delete their account;
  - report their departure and arrival times;
  - view their dashboards.
- ✓ Managers features:
  - manage their team(s);
  - view the averages of the daily and weekly hours of the team over a given period;
  - view the daily and weekly working hours of an employee over a period of time;
  - view their employees' dashboards.

This list of features is non-exhaustive (no more than the list of dashboards).

It is essential to adapt it according to your research and the various audits that you will carry out, so that the application corresponds to the needs of each category of users.

KPI

You must choose at least 2 types of KPIs (e.g., lateness rate, etc.).

Authentification

User Management

As the application is centred on the digital life of users, it must therefore offer a management of the latter.
To do this, you must create a user management module.
The client asks non-identified users to be registered by the managers.
An administration section would be useful to manage site users.

Request Authentification

Each API request must be **secured with a token** when the page requires authentication.

Json Web Token

Authentication / Identification

Using the application requires knowing the user in question.
To do this, it is necessary to implement the following option:

- ✓ a method of user authentication via a username / password;
- ✓ password management

Fake email, such as Mailpit.

Ensure that basic security measures are in place to prevent credential theft.

Technologies backend

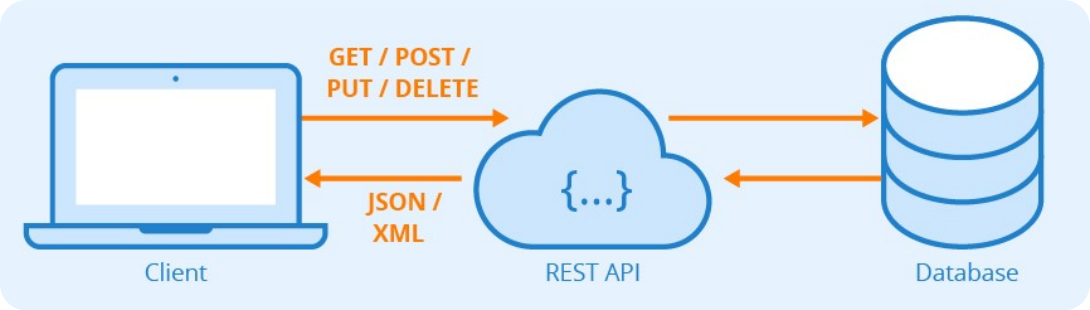
You can choose (and justify why) from the following technologies to build your project :

- ✓ **JS**: Node (express is allowed);
- ✓ **Compiled**: Go;
- ✓ **Python** : Django;
- ✓ **PHP** : Symfony;
- ✓ **Java** : Spring boot;
- ✓ **Elixir** : Phoenix;
- ✓ **SQL**: MariaDB or POSTGRESQL;
- ✓ **NoSQL**: MongoDB.

Anything not on this list is prohibited.

Frontend application

To develop the frontend, you have two options: build a mobile application or a web application.
You must use the RESTFUL API that was developed to implement the frontend.



Interfaces

You must implement, at least, the corresponding views:

- ✓ Authentification:
  - login;
- ✓ Users:
  - retrieves the users;
  - adds a user;
  - updates a user;
  - deletes a user;
- ✓ Teams:
  - retrieves the teams;
  - adds a team;
  - updates a team;
  - deletes a team;
- ✓ Time management:
  - set the arrival / departure of the authenticated user;
  - set a summary of the arrivals and departures of an employee;
  - get arrivals and departures;
- ✓ KPI:
  - get a global report based on the chosen KPIs.

Role and security

There are two roles, namely **employee** and **manager**.
It is important to manage security on the pages according to the respective roles.

Of course, the requests must be protected to restrict access only to those who are authorized.
Using the application requires knowing the user in question.

To do this, it is necessary to implement the following options:

- ✓ a method of user authentication;
- ✓ password management;
- ✓ token.

Below is a list of features that you should implement:

- ✓ Common features:
  - edit their account informations;
  - delete their account;
  - report their departure and arrival times;
  - view their dashboards.
- ✓ Managers features:
  - manage their team(s);
  - view the averages of the daily and weekly hours of the team over a given period;
  - view the daily and weekly working hours of an employee over a period of time;
  - view their employees' dashboards.

This list of features is non-exhaustive (no more than the list of dashboards).
You can add more if you find them relevant.

UX & Accessibility Awareness

To ensure your application can be effectively used by all users, regardless of their context, device, or abilities, you must integrate good UX (User Experience) and Accessibility practices throughout your project.

Building an accessible and usable interface is not just a matter of compliance, it's a matter of quality, inclusion, and efficiency.

Why it matters

A clear and consistent user experience increases adoption and reduces user errors.

Accessibility ensures that people with disabilities (visual, motor, hearing, cognitive) can use your product.

It improves SEO, usability, and overall quality perception.

Your project must include accessibility and UX considerations from the design phase to the CI pipeline.

a11y

Technologies frontend

You can choose (and justify why) from the following technologies to build your project :

- ✓ **Frontend**: React.js, Vue.js, Angular;
- ✓ **Android application**: Kotlin, Java;
- ✓ **iOS application**: Swift;
- ✓ **Cross Platform**: React Native, Flutter.

Anything not on this list is prohibited.

Documentation

Take the time to define a simple and scalable architecture, without code duplication.

You are expected to provide clear and simple documentation of your project.

The goal is to have a document that serves as a working support to easily understand the project in order to facilitate communication in the work teams and facilitate the integration of new developers.

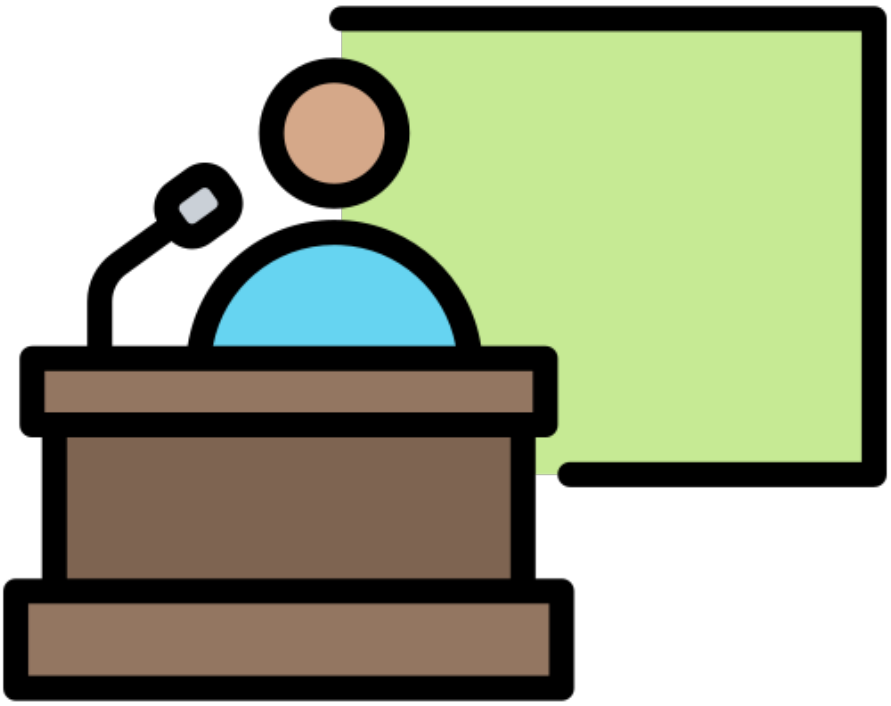
Thus, there is no need to make class or sequence diagrams of the entire project, but rather to choose the important parts to understand what needs to be documented.

You must provide a README.md file describing your project and how setup it for the launch. This file must be written using the [markdown format].

Keynote

At the end of your project you will have to:

- ✓ present the good practices that you have implemented to build your project;
- ✓ present your documentation;
- ✓ show your CI in action;
- ✓ defend your technos' choice;
- ✓ show your code and architecture.



v 1.3