REPUBLIC OF TUNISIA
MINISTRY OF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH
UNIVERSITY OF CARTHAGE
FACULTY OF SCIENCES OF BIZERTE

SUMMER INTERNSHIP REPORT
SECOND YEAR OF ENGINEERING SCHOOL
IN SOFTWARE ENGINEERING

By :

Baha Eddine BRAHIM

development of a module for taking online tests

	Mr.Soltani Mondher
professional supervisor:	
technical supervisor:	**Ms.Mejri Khouloud**
	Ms.Zribi Sirine

Fullfilled in

Année Universitaire: 2023 - 2024

ACKNOWLEDGEMENT

It is with great honor that I write these lines to thank all those who contributed directly or indirectly to the realization of this project.

My thanks go first to my supervisors, **Ms. Mejri Khouloud** and **Ms. Zribi Sirine**, for their help and availability, as well as for their assistance and advice.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

HTML HyperText Markup Language

JS JavaScript

CSS Cascading Style Sheets

API Application Programming Interface

XML eXtensible Markup Language

JSON JavaScript Object Notation

HTTP HyperText Transfer Protocol

UML Unified Modeling Language

SQL Structured Query Language

MVVM Model View View Model

MVC Model-View-Controller

REST Representational State Transfer

URI Unified Resource Identifier

I

General presentation of the project

Introduction

In this chapter, we will discuss the general context in which my project of a QUIZ module of a CVThèque web application fits, as well as the objectives that we have set to meet the identified needs.

I.1 Presentation of the host organization

L'objectif de cette section est de fournir un aperçu du outlYing SUARL, et de son architecture organisationnelle.

I.1.1 Context

Young Tunisian startup, specialized in information technologies. outlYing works on different themes related to digital and digital transition. Launched by Mr.Mondher SOLTANI to allow young Tunisian talents to integrate the world of work by supporting companies in their digital transitions.

I.1.2 organisation Structure

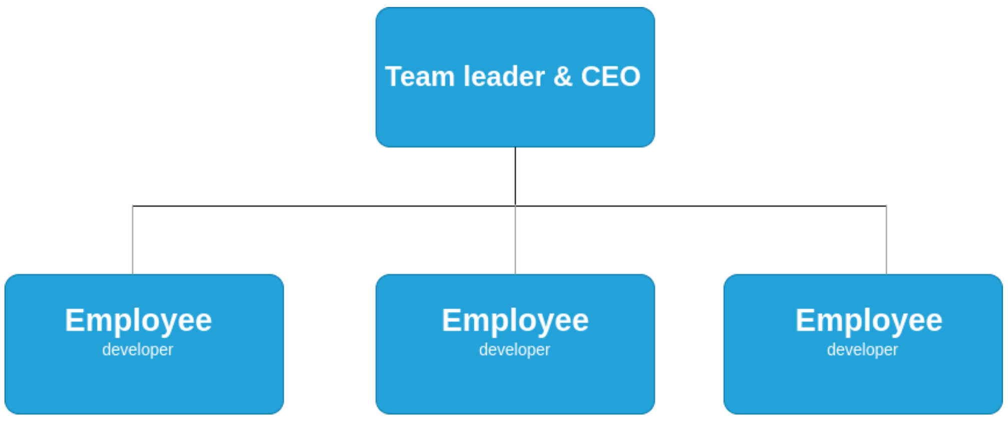


Figure I.1: General structure of OutlYing

I.1.3 Analysis of the existing situation

In order to facilitate access to the job market for Tunisian talents who have computer skills and expertise without necessarily traveling, outlYing has decided to develop an application that allows users to authenticate and create online resumes by marking their different skills and it will help companies to find talents easily. So outlYing must verify the competence of users according to their level declared in their CVs.

I.1.4 Proposed Solution

The main goal is:

– Offering the users a chance to prove their skills with a test en ligne using Quizzes.

I.2 Development methodology

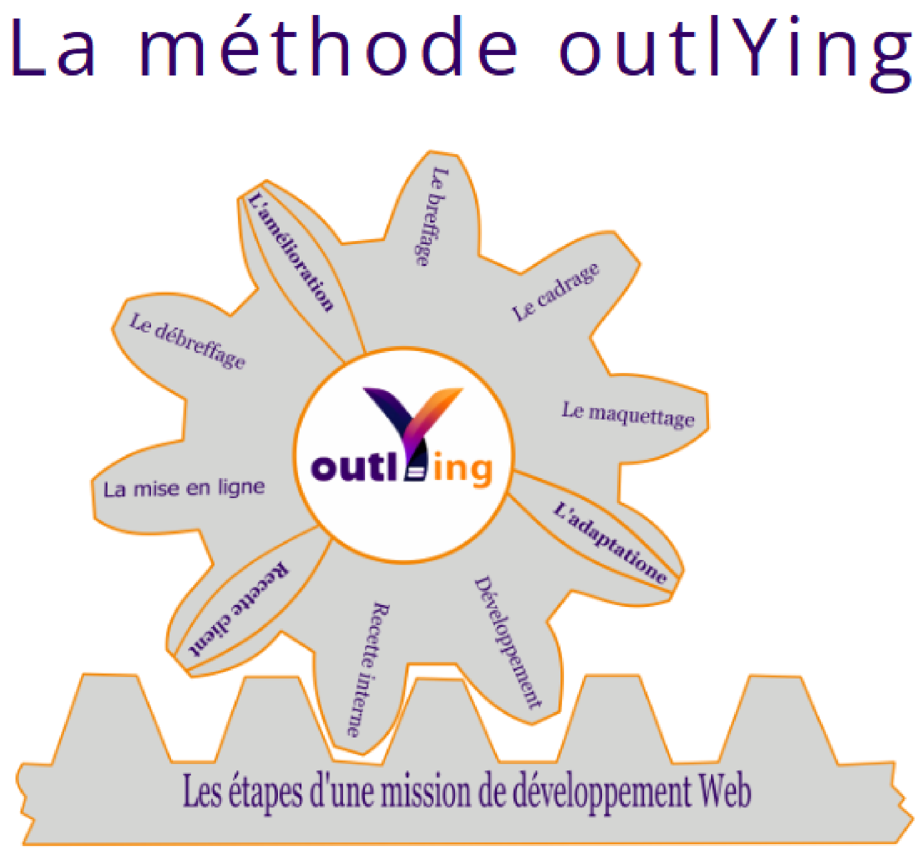


Figure I.2: les méthodologies de travail

Conclusion

Throughout this chapter, I introduced OutlYing SUARL by giving an overview of its background, its organizational chart . After that, we analyzed the state of art with the purpose of determining the client's needs and then presenting our proposed solution. . In the second chapter, we will identify our system's requirements, the respected architecture .

II

Architecture and planification

Introduction

In this chapter, we will identify the main actors, functional and non-functional specifications, the global use case diagram and the adopted architecture designs. And at last, we will give a general overview of the used technologies which will be detailingly explained in the annexe.

II.1 Requirement Analysis

II.1.1 Actors identification

We have a hierarchy type of user, first is the a user who hust logged in. Second the one who entered his CV , and last the one who is passing test who is called "Esprit Brillant"

II.1.2 Functional requirements

Functional requirements represent the functionalities that should be provided by the produced software.
Our solution shall enables its client to:

- The user choose his own quiz test.
- Capability to navigate between all the quiz questions .
- Capability to consult the score after the test.

II.1.3 Non-Functional requirements

- **Performance:** The resulting product must reaction quickly to the users demands and provides fast pages refreshing.
- **Maintainability:** In order to facilitate changing or adding something to the platform, coding must follows specific standards.

II.2 Use case diagram

The use case diagram is a depiction of the user's interaction with the developed system each use case represents a use of the system. This diagram is the simplest way to define the functional requirement of the project. The represented diagram is the global use case diagram for our proposed solution.

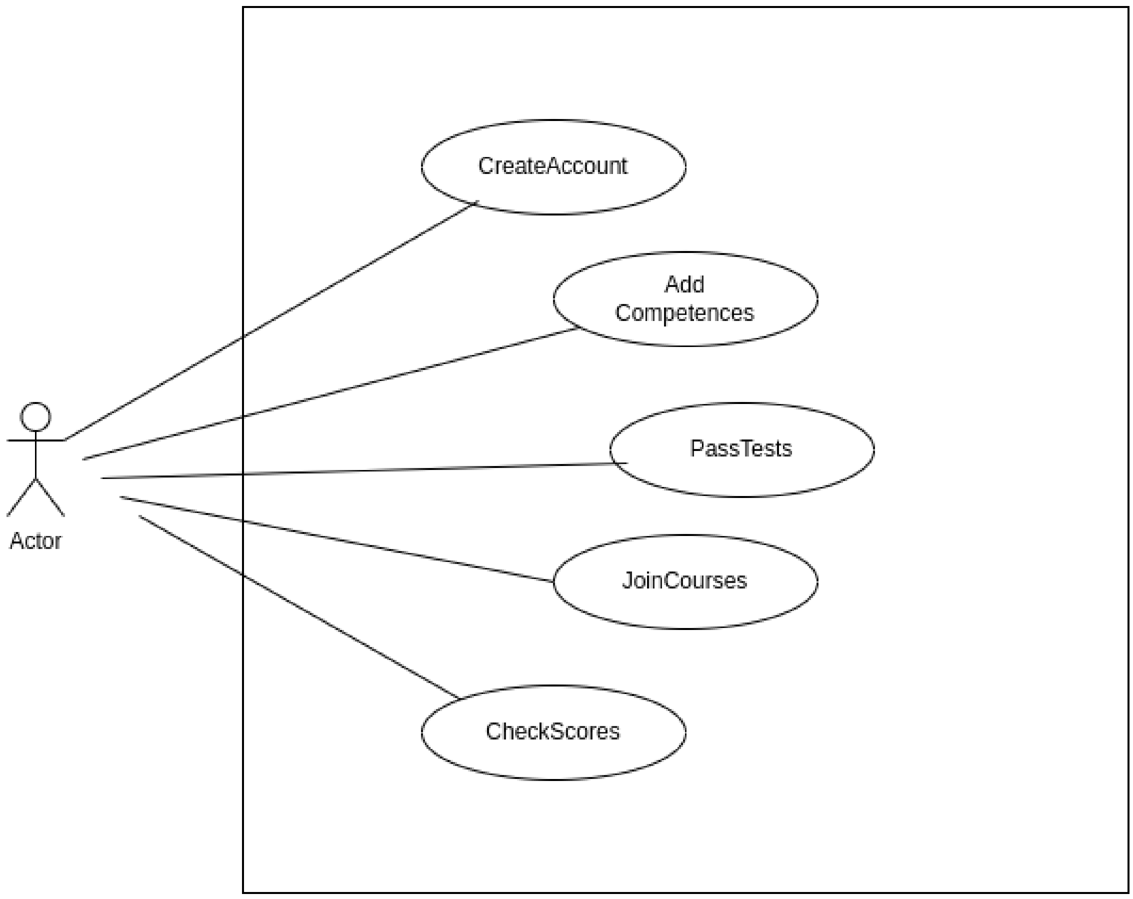


Figure II.1: Global use case diagram

In our case, we will focus only on the passTest case only

II.3 Class diagram

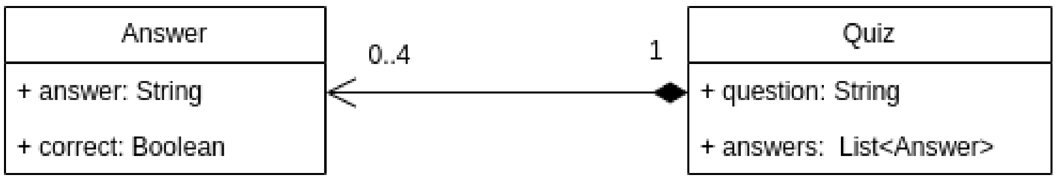


Figure II.2: Class diagram of the first sprint of the first release

II.4 REST pattern for web API

Representational State Transfer is an architecture style for constructing distributed systems. It was introduced by Ray Fielding as a sum of constraints. REST is not strictly based on HTTP, but it is most commonly related to it. Over the years, it became commonly known model for the web Application Programming Interface(API). Every web service respecting the REST pattern is called RESTful API. In this section, we will give a detailed explanation of REST pattern and web services starting with few definitions.

- **Resource:** A resource is like an entity in a database that has a unique name and accessed through at least one URI. It can be for example an account. Also a group of resources is called Collection.
- **URIs:** Unified Resource Identifiers are, as they are named, identifiers of the resources. They are used for the access to the resources.
- **HTTP:** HyperText Transfer Protocol is a used protocol by the World Wide Web defining the way of transmitting and formatting messages. It is known for being stateless which means that executed commands are independent from the previous ones.
- **Endpoint:** It is an unique URL representing a resource or a collection of resources.
- **API:** Application Programming Interface is a description of the way of interaction between different software components. It is the responsible part in the server of receiving requests and sending responses.

II.4.1 CRUD methods

As we mentioned earlier the REST pattern is related to HTTP protocol. Therefore, it uses the CRUD methods provided by the HTTP. The table down below gives an explanation of these methods and their use.

Table II.1: HTTP CRUD methods

HTTP methods (verbs)	CRUD actions	Description	Example
Create	POST	Resource creation	http://xyz.com/Account (we have created a new account).
Retrieve	GET	Finding a resource	http://xyz.com/Account/4582 (finding an account with the ID 4582) or http://xyz.com/Accounts (finding a collection of accounts)
Update	PUT	updating a resource	http://xyz.com/Account/4582 (updating the account with the ID 4582)
Delete	DELETE	removing a resource	http://xyz.com/Account/7135 (removing an account with the ID 71350)

II.4.2 HTTP status codes

Also the HTTP have status codes which are also a standard indicates the result of an HTTP request and the next tables shows the used ones in the project.

Table II.2: The HTTP status codes

Status Code	Meaning	Description
200	OK	The requested data, by the consumer, from the server was found.
201	Created	The consumer gave data the server in order to create a resource.
204	No content	The server removed the resource given by the consumer.
401	Unauthorized	The client is unauthorized to access to the demanded resource he needs to authenticate.
403	Forbidden	Though the server understood the request it did no action for that the client don't have the permission for that resource.
404	Not found	the server informs The consumer referenced Resource or Collection is nonexistent.
405	Method not allowed	The requested resource doens't support the request method.

II.4.3 RESTful API structure

After understanding the REST pattern principals, giving a detailed example of a RESTful API structure would be the best way to understand our API structure and this is what the image down below is all about.

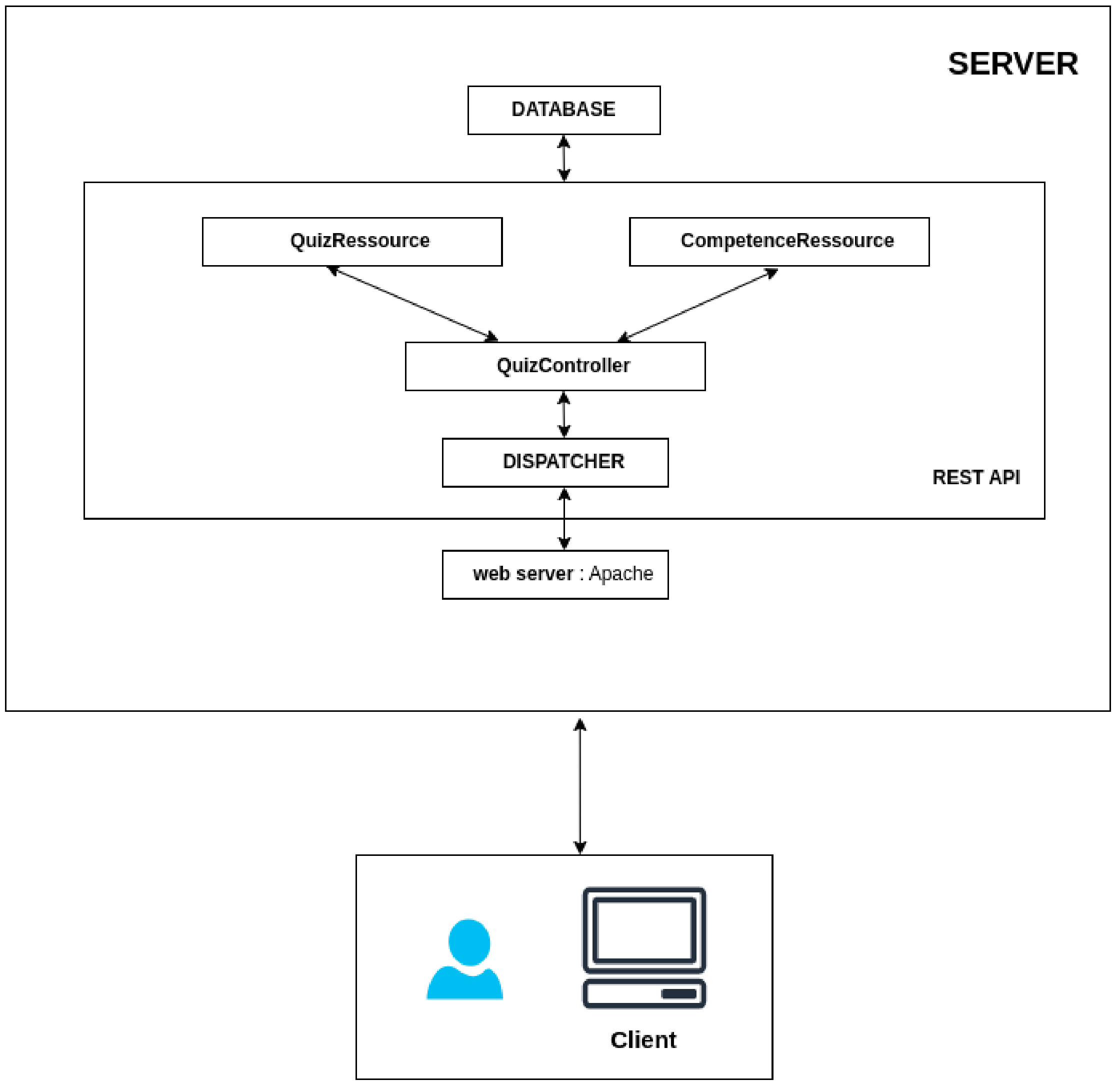


Figure II.3: RESTful API structure

II.4.4 Advantages of REST over SOAP

To invoke web services, REST pattern is not the only available one for such a mission. There is also SOAP, Simple Object Access Protocol, which is a standard communication protocol specified for XML-based message exchange. However, we chose to use the REST pattern for our API because of the advantages it provides over SOAP which are:

- REST is known for being Stateless which means that the server does not store on its side any state about the client session. However this latter is stored on the client side and passed to the server only as needed.
- REST is easily understood, documented and written which make writing APIs very

simple and since REST is concentrating on resource-based HTTP operations, web-browsers and consumers find it easy to consume (extremely beneficial for public APIs).

- REST is very similar to other Web technologies design and provides requests entirely independent of each other, therefore, it is considered quicker than SOAP. It enables fast calls to a URL for quick return responses.
- Unlike SOAP which is restricted to the use of only XML for all its messages, REST is a lightweight choice for Web service access, it can use smaller messages format like JSON.

II.5 MVVM pattern

For the front-end side we chose to use Angular2 which follow the MVVM pattern. This latter, the Model-View-View Model, is a software architectural pattern extended from the MVC pattern. Therefore, it is considered as a super-set of the MVC or an improvement of it.It is made for the aim of enabling a real separation between the presentation layer (View) and the logic layer(Model) and providing the "two way binding" mechanism.

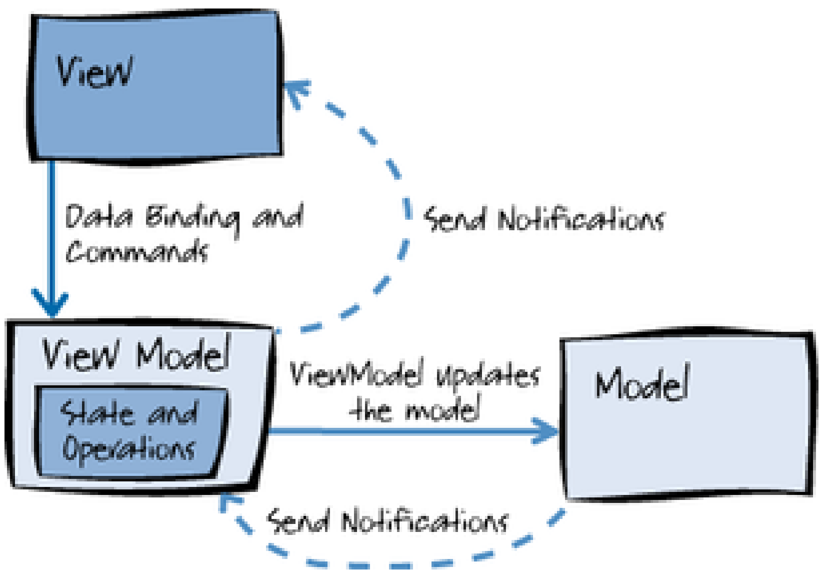


Figure II.4: Model-View-View Model

II.5.1 Model

The model in the MVVM pattern is the data holder. It is responsible for managing data and updating it.

II.5.2 View

represents the structure, layout, and appearance of what is seen by the user on the screen. A view gets data from its view model or invokes methods on the view model through the 2-way binding.

II.5.3 View Model

View Model is a JavaScript function which is responsible for maintaining relationship between view and model, but the difference between it and the Controller in the MVC pattern is, if we update anything in view, it gets updated in model and vice-versa which is what we call two-way binding.

II.5.4 Advantages of the MVVM pattern

As we have mentioned the MVVM was made as improvement for the MVC pattern. So obviously it has extra advantages over the MVC pattern which are:

- Better separation of layers: In the MVC pattern we always find ourselves in a situation wondering which code belong to what. And since every code that doesn't belong to neither the view or the model is putting in the controller, we end up having a fat, hardly tested and managed controllers. However, in the MVVM pattern a view model was added to the mix and relieved the controller from the task of translating the model data into something usable by the view.

- Improved Testability: By adding the view model, controller is no longer dependent to the model layer so it became much easier to test it. Since some of the controller's mission were taken by the view model view controllers is lighter and easily tested.

- Transparent Communication: The responsibilities of the view controller are lessen to controlling the interaction between the view layer and the model layer,to put it in short, gluing both layers together. The view model provides a transparent interface to the view controller, which it uses to populate the view layer and interact with the model layer. This results in a transparent communication between the four layers of your application.

II.6 Working environment

The XP methodology gives a huge importance to coding. Thus, in the next paragraph we will give a description to the used development tools throughout the development process.

II.6.1 Software environment

Here, we listed the used software throughout the process of our project development:

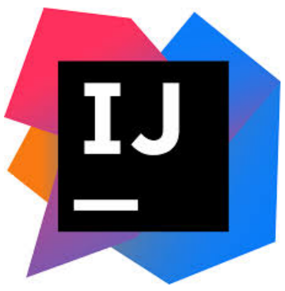


Figure II.5: IntelliJ IDEA Ultimate

- **IntelliJ** IDEA Ultimate offre un support de premier ordre pour les principaux frameworks et technologies destinés au développement d'applications et de microservices modernes. Votre Il est fourni avec une assistance dédiée pour Spring et Spring Boot, Jakarta EE, JPA, Reactor et d'autres frameworks.

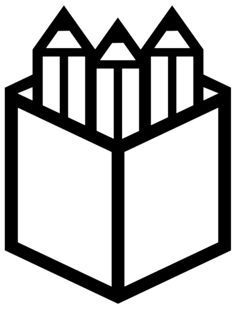


Figure II.6: Penpot

- **Penpot** est l'outil de conception open source basé sur le Web qui comble le fossé entre les concepteurs et les développeurs .

Figure II.7: Microsoft Visual Code

- **Visual Studio Code:** It is a source code free and open-source editor developed by Microsoft. It provides a powerful developer tooling like debugging, intelligent code completion,....

Figure II.8: PostgreSQL

- **PostgreSQL** est un système de gestion de base de données relationnelle orienté objet puissant et open source qui est capable de prendre en charge en toute sécurité les charges de travail de données les plus complexes

II.6.2 Technologies

Every project creation is based on specific technologies including programming languages and frameworks. And for our project we tried to use the most adequate, and recent, technologies to reach perfection in our resulting product. These technologies are listed below and they will be detailingly explained in more detail in the annexe:

- **Angular 17:** It is a framework for building client applications in HTML and JavaScript or any other language that compiles to JavaScript like the TypeScript language that we used in our project.
- **TypeScript:** Since we are using the Angular2 framework for the front-end side we chose to use as a scripting language the TypeScript language which is a free and open-source programming language developed and maintained by Microsoft.
- **JSON:** A shortened form of JavaScript Object Notation. It is a lightweight data interchange and storage format. It provides a readable and well organized information it is based on a subset of the JS language.
- **LaTeX:** it is a document preparation system for high-quality typesetting.[?]

Conclusion

In this chapter, we identified the software requirements and the functional and nonfunctional specifications. Then, we presented the global use case diagram as well as a the backlog project and defined the sprints within the releases. Finally, we mentioned the adopted architectures and the used technologies. Clearly, this chapter contains a complete specification for our proposed solution it enormously helps us to structure our proposed solution in an understandable way this phase leads to path to the conception phase which will be exhaustively presented and explained in the next chapter.

III

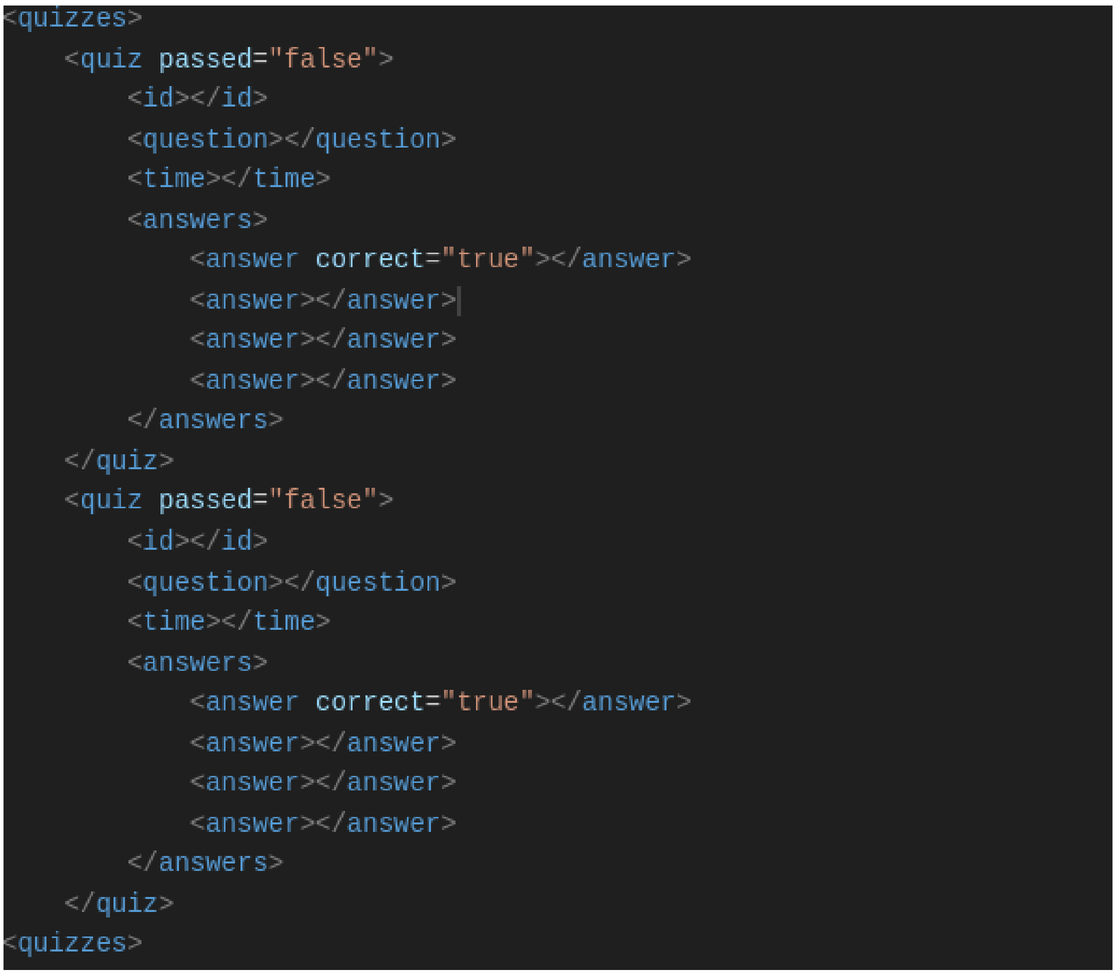
Release

III.1 Conception

The section focuses on the "conception" and diagrams.

III.1.1 XML file structure

we will use an XML file for retrieving quzzes : question and answers



```
<quizzes>
    <quiz passed="false">
        <id></id>
        <question></question>
        <time></time>
        <answers>
            <answer correct="true"></answer>
            <answer></answer>
            <answer></answer>
            <answer></answer>
        </answers>
    </quiz>
    <quiz passed="false">
        <id></id>
        <question></question>
        <time></time>
        <answers>
            <answer correct="true"></answer>
            <answer></answer>
            <answer></answer>
            <answer></answer>
        </answers>
    </quiz>
<quizzes>
```

Figure III.1: XML file structure

III.1.2 Business logic of Quiz mmodule

– The brilliant mind can take only one test for each skill (unless the manager allows him to repeat).

– Each test (of 25 quizzes) will have 5 difficulty levels, so 5 questions for each level.

- The quizzes of the second test must not have the same questions as the first test (min 80 % new quizzes)
- you must have a range of quizzes: for each level you must have for example 10 quizzes, we will choose 5 randomly.
- Timing: An overall time duration will be granted for the entire test (eg: 5 min total and not 10s for each quiz).
- During the test, the brilliant mind can navigate between the quizzes and change his answers (so you must indicate the number of quizzes that the brilliant mind has not answered out of the total number).
- After the test: The brilliant mind cannot see the false and true answers , Skill boxes will have an update or they are filtered.

III.1.3 Use Case Diagram

this diagram represents all the use cases of all the modules , but we will focus next only in passTest use case

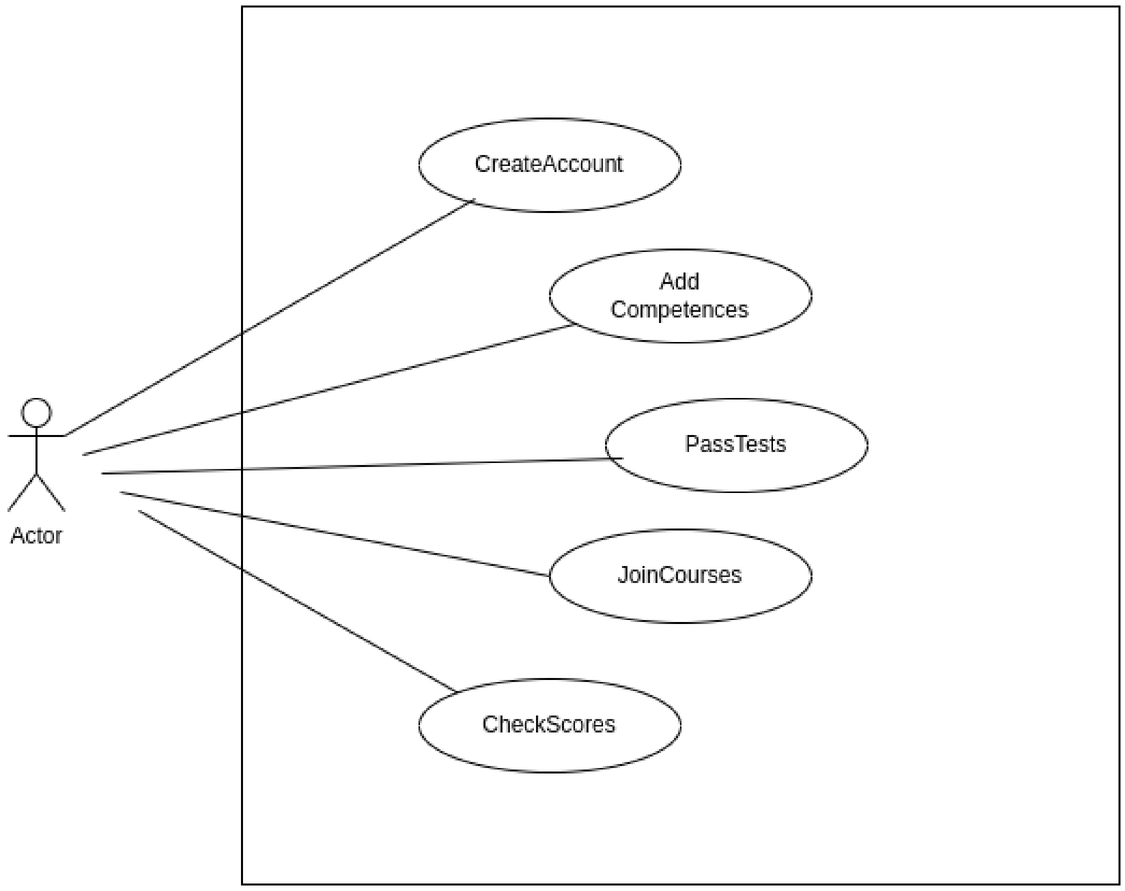


Figure III.2: CVtheque UseCase diagram

III.1.4 Textual description of QUIZ scenario

Table III.1: Textual Description

Use case	Take a quiz test
Actor	• User
Optimal scenario	• The user reads the introduction of the first pop-up and clicks on “Next”. • They read the message displayed in the second pop-up and click on “Start”. • They navigate between the quizzes, choosing and changing answers. • They finish the test by clicking on “Finish”.
Pre-condition	• Permission granted by the manager. • Skill box must be enabled. • User clicks on the skill box.
Post-condition	• User clicks on the "Finish" button.

III.1.5 Quiz catalog conception

the four categories are devided on two sections:

- Cognitive and behavioral skills :
  - All the quizzes are the same to all users
  - Are accessible through the dashboard or through the profile

- Technical and sectoral skills
  - The uer will be redirected through to the convenient Quiz .

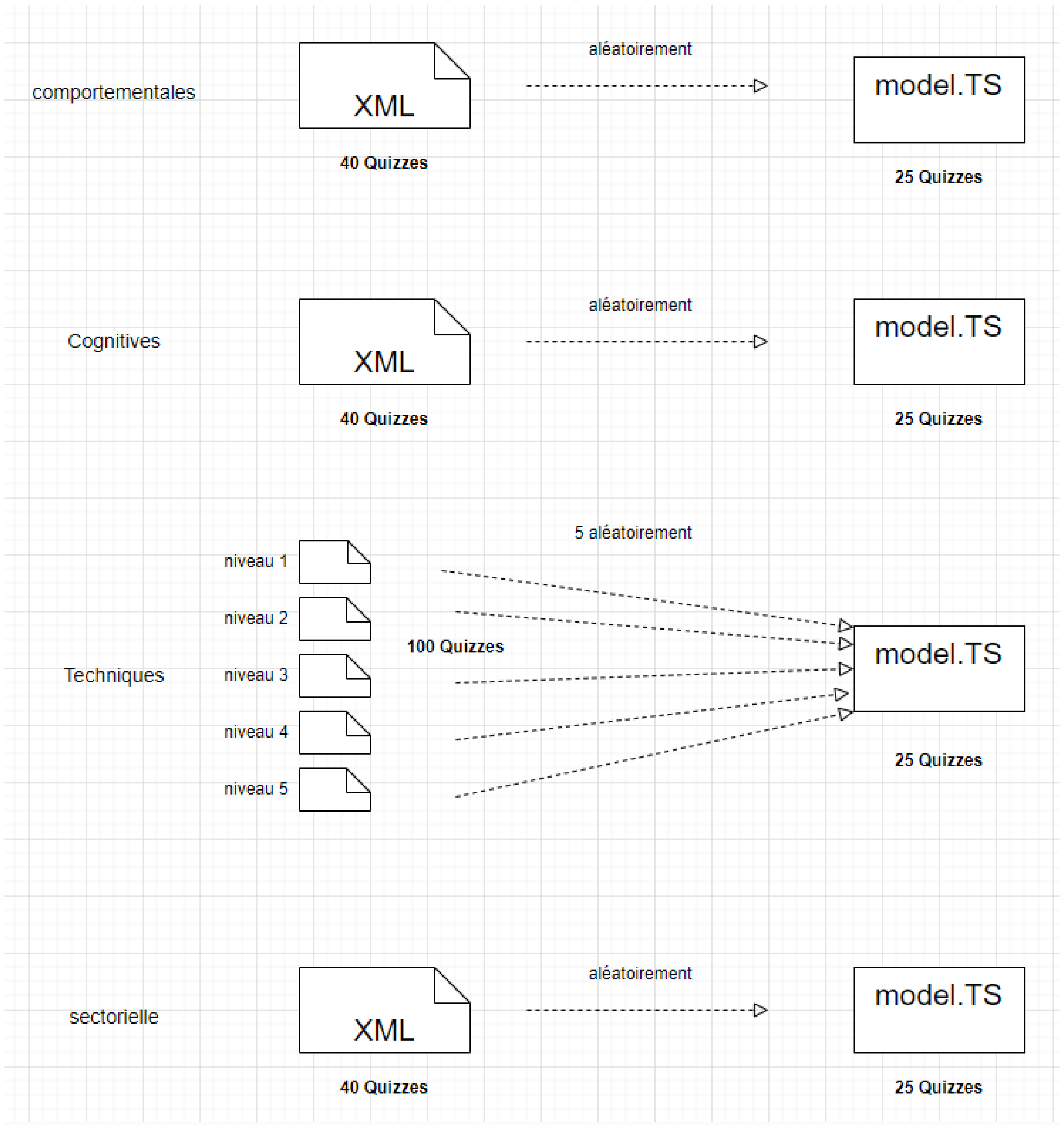


Figure III.3: Quizzes Distribution

III.1.6 Quiz management

the user will have a group of quizzes regarding four categories :

- Cognitive and behavioral skills :

  - All users will take the same (common) tests, as soon as they click on the link, they will be redirected to the test page.

- Technical skills
  - Les tests seront filtrés selon les informations remplies par les utilisateurs(dans la page esprit) .

- Sectoral skills
  - An interface will be displayed to users containing all the tests and users can do a search (either by writing in the search bar or with a filter list) (we will give the user the choice because he may be interested in another domain (example a fullstack who works in the banking domain). .

III.1.7 Quiz details

- The date and time will be recorded in the database in the appropriate table (see score tables).
- The user will have in his dashboard or profile a list of his accomplishments, also a link to the history of all his tests. If we click, a page opens that contains a table with columns: skill name, score, date, successful or not. (with a search bar above the table).

III.1.8 Class diagram

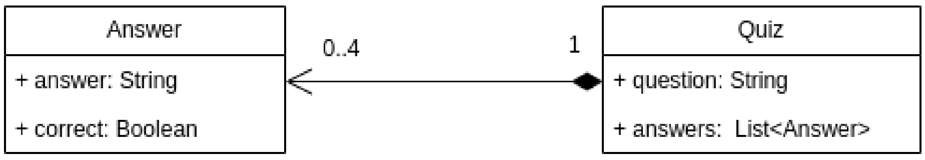


Figure III.4: CVtheque UseCase diagram

III.1.9 Seq diagram

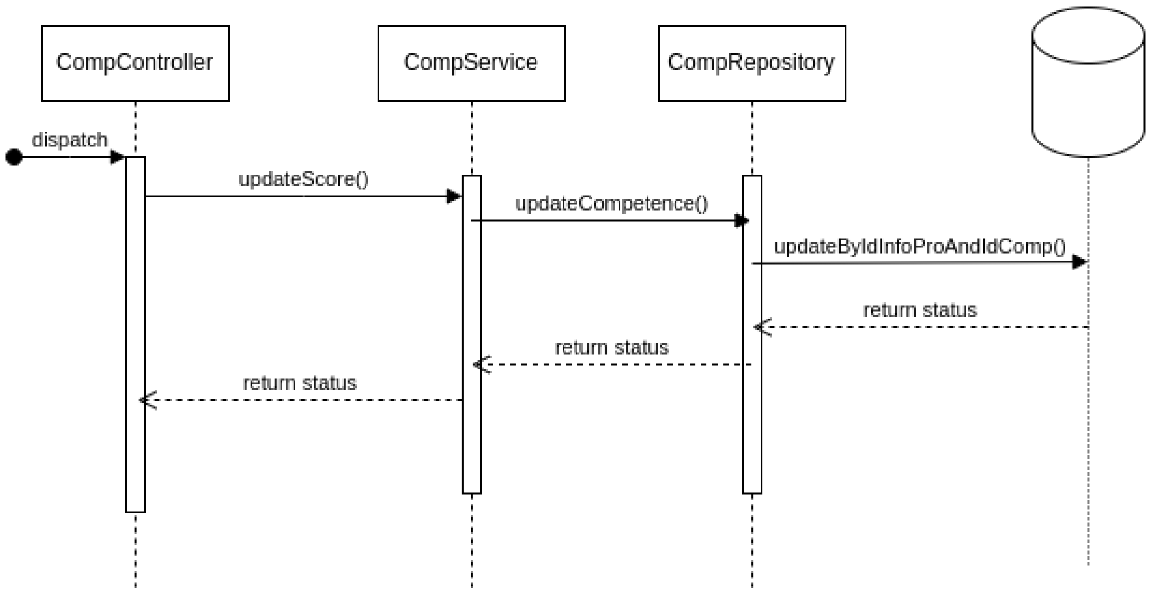
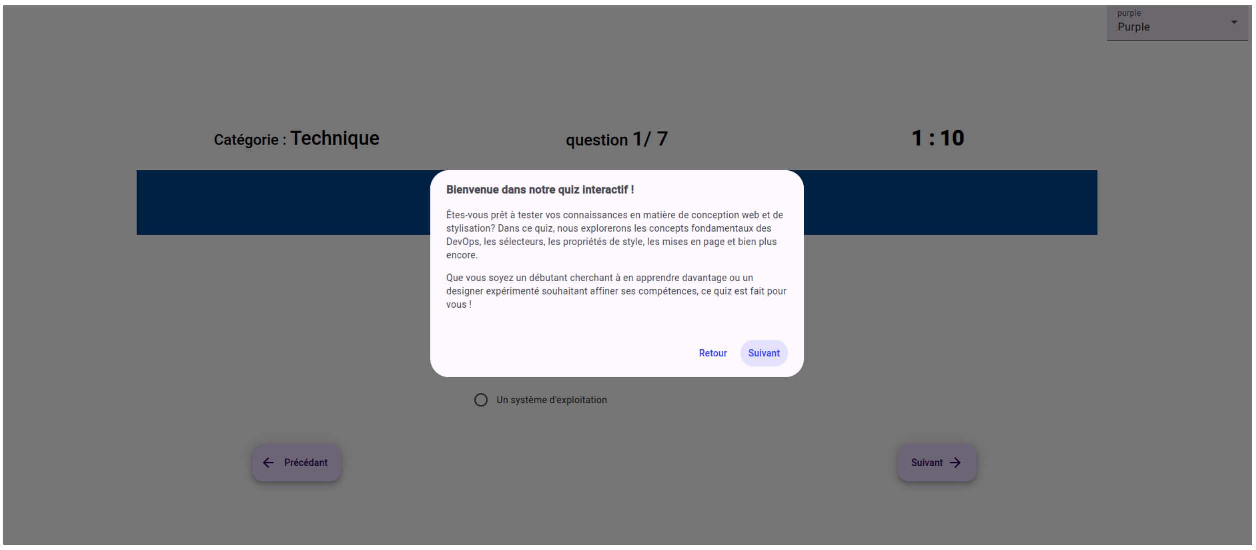


Figure III.5: Adding / updating score

III.2 Interfaces

In this section, we present screenshots of the resulting product from the Quiz test.

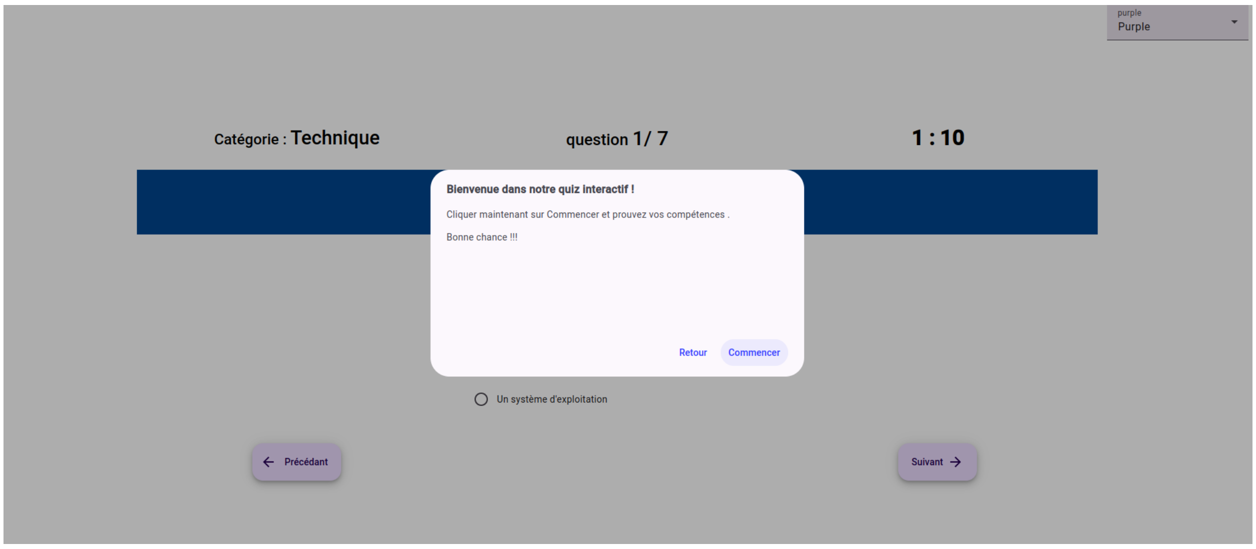


Bienvenue dans notre quiz interactif !

Êtes-vous prêt à tester vos connaissances en matière de conception web et de stylisation? Dans ce quiz, nous explorerons les concepts fondamentaux des DevOps, les sélecteurs, les propriétés de style, les mises en page et bien plus encore.

Que vous soyez un débutant cherchant à en apprendre davantage ou un designer expérimenté souhaitant affiner ses compétences, ce quiz est fait pour vous !

Figure III.6: First PopUp as introduction



Catégorie : Technique

question 1 / 7

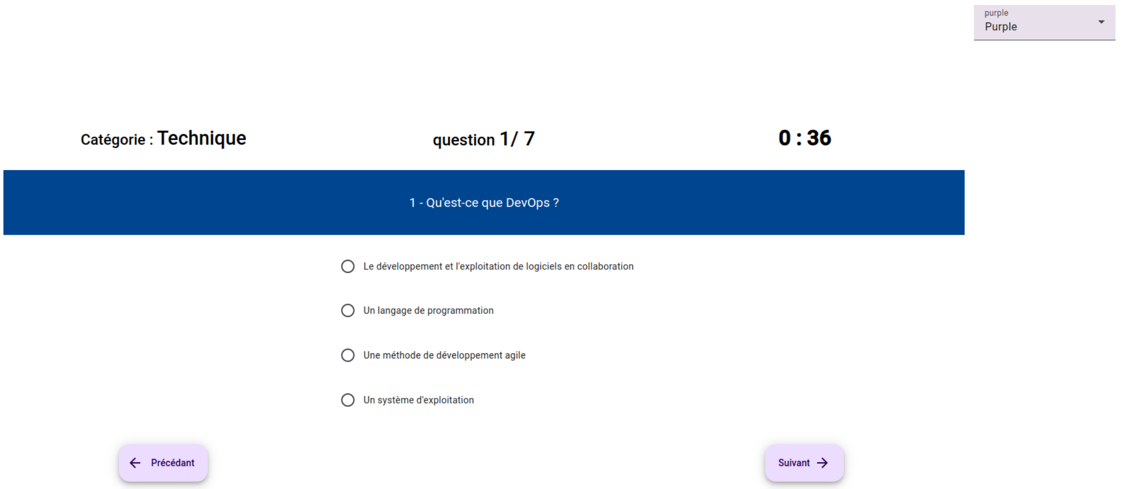
1 : 10

Bienvenue dans notre quiz interactif !

Cliquer maintenant sur Commencer et prouvez vos compétences .

Bonne chance !!!

Figure III.7: warning before entering



Catégorie : Technique

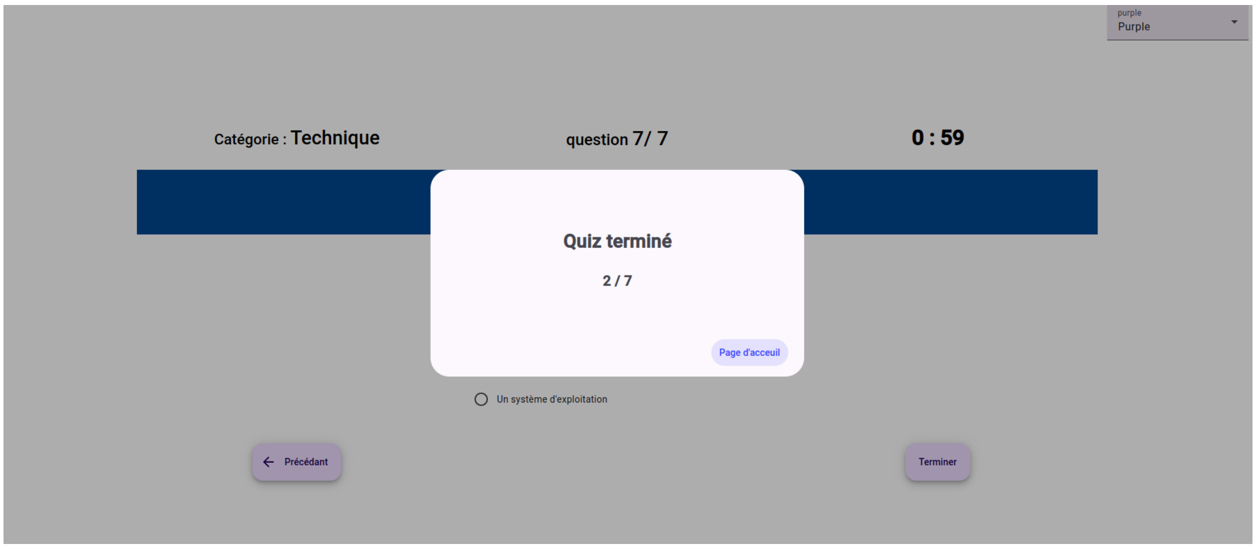
question 1 / 7

0 : 36

1 - Qu'est-ce que DevOps ?

- Le développement et l'exploitation de logiciels en collaboration
- Un langage de programmation
- Une méthode de développement agile
- Un système d'exploitation

Figure III.8: During the test



Catégorie : Technique

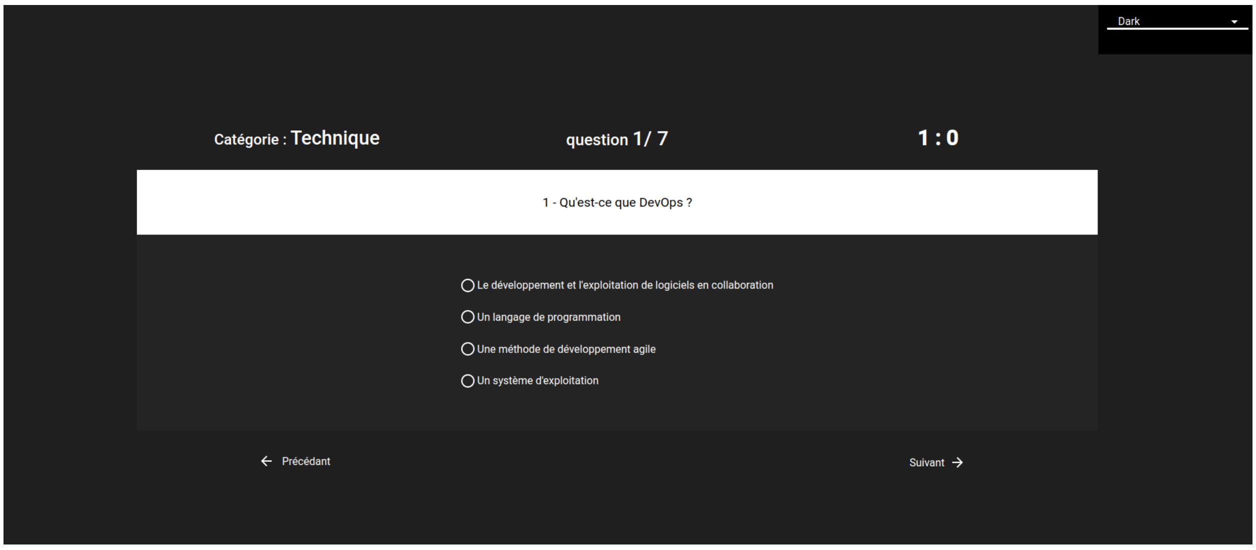
question 7/ 7

0 : 59

Quiz terminé

2 / 7

Figure III.9: Score after terminating the quiz



Catégorie : Technique

question 1/ 7

1 : 0

1 - Qu'est-ce que DevOps ?

- Le développement et l'exploitation de logiciels en collaboration
- Un langage de programmation
- Une méthode de développement agile
- Un système d'exploitation

Figure III.10: Dark mode

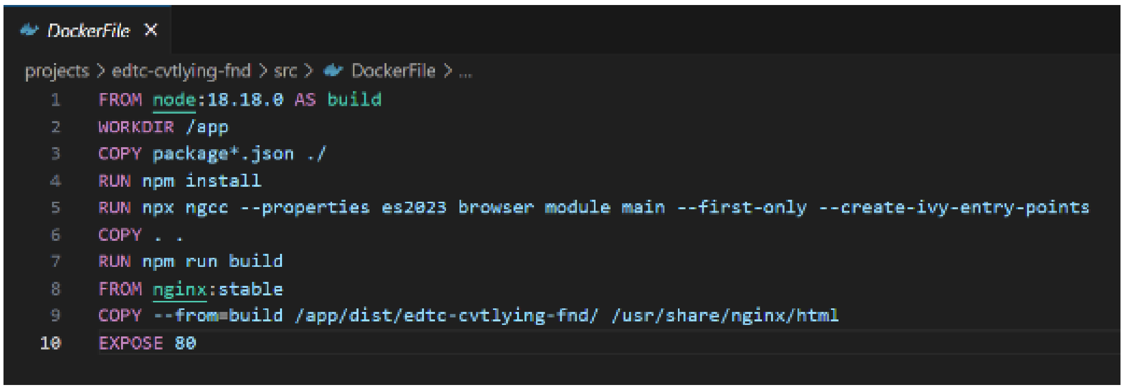
Conclusion

In this chapter, I presented interfaces of all the phases of the quiz test with different themes
.

III.3 Local deployment

In this section, I will present screenshots of the deployment procedure of the project .

III.3.1 Dockerizing the front end - angular



DockerFile ×

projects > edtc-cvtlying-fnd > src > DockerFile > ...

```
FROM node:18.18.0 AS build
WORKDIR /app
COPY package*.json ./
RUN npm install
RUN npx ngcc --properties es2023 browser module main --first-only --create-ivy-entry-points
COPY . .
RUN npm run build
FROM nginx:stable
COPY --from=build /app/dist/edtc-cvtlying-fnd/ /usr/share/nginx/html
EXPOSE 80
```

Figure III.11: docker-file

III.3.2 Explanation

- FROM node:18.18.0 AS build : Sets the base image to use for the first build step. We use node:18.18.0 as the base.
- WORKDIR /app : Sets the working directory in the Docker image as /app.
- COPY package*.json ./ : Copies the package.json and package-lock.json files to the /app working directory.
- RUN npm install : Installs the dependencies defined in package.json.
- RUN npx ngcc –properties es2023 browser module main –first-only –create-ivy-entry-points : Runs the Angular Ivy compiler (ngcc) to prepare dependencies for running in an Ivy environment (Angular's new rendering engine). => This is recommended for production performance.
- COPY . . : Copies all files from the local source directory to the /app working directory of the Docker image.

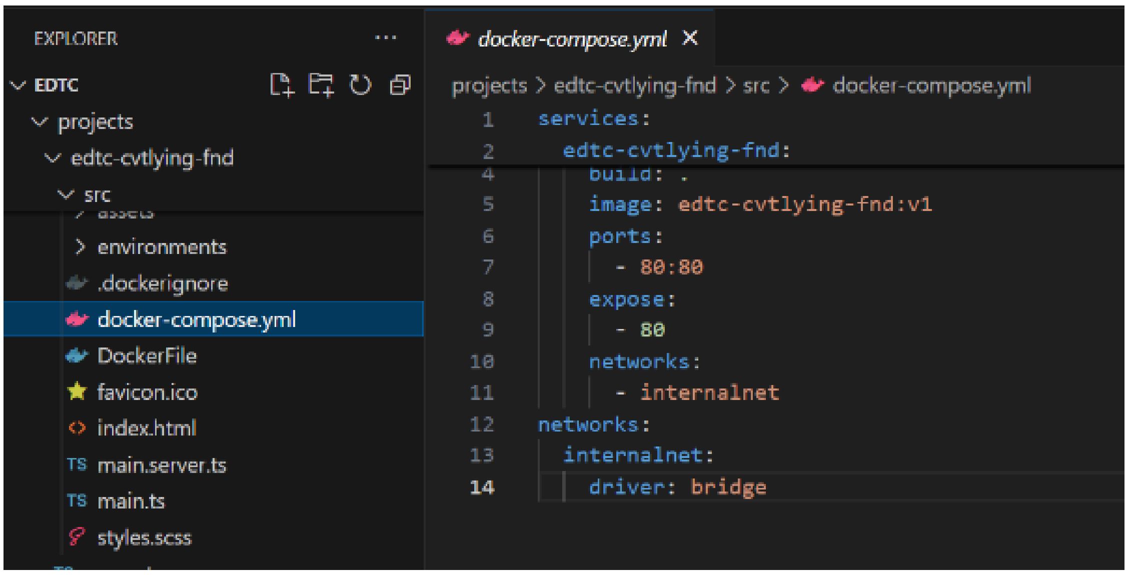
- RUN npm run build : Builds the Angular application using the command defined in package.json to build the application. => Typically, this creates a dist directory containing the production files of the application.

- FROM nginx:stable : Uses nginx:stable as the base image for the second stage of the Docker image. Nginx is used here as the HTTP server to serve the static frontend files.

- COPY –from=build /app/dist/edtc-cvtlying-fnd/ /usr/share/nginx/html : Copies the Angular application build files from the previous stage (build) to the /usr/share/nginx/html directory of the nginx image. This configures Nginx to serve these static files when the container is running.

- EXPOSE 80 : Exposes port 80, the default port for HTTP, to allow access to the contents served by Nginx.

III.3.3 Docker-Compose



```yaml
services:
  edtc-cvtlying-fnd:
    build: .
    image: edtc-cvtlying-fnd:v1
    ports:
      - 80:80
    expose:
      - 80
    networks:
      - internalnet
networks:
  internalnet:
    driver: bridge
```

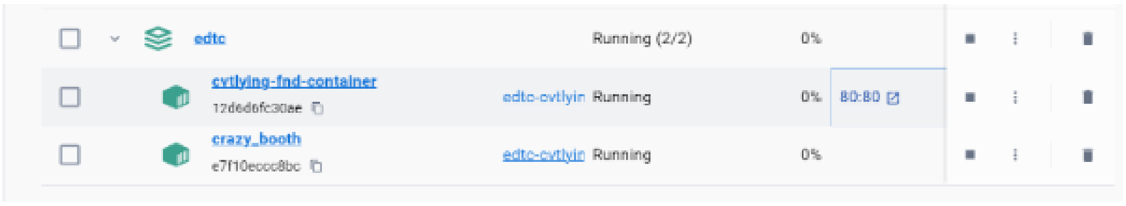
Figure III.12: docker-compose

III.3.4 Explanation

- edtc-cvtlying-fnd : This is the name of the Docker service that represents the service for the frontend application.

- containername : Defines the name of the Docker container that will be created from the image.
- build: . : Specifies the path to the Dockerfile used to build the service image. Here, "." means that the Dockerfile is in the current directory.
- image: edtc-cvtlying-fnd:v1 : The name of the Docker image to use for this service, if it does not exist locally, Docker will build it using the specified Dockerfile.
- ports : Maps the container ports to the host machine ports. In this example, - 80:80 means that the container port 80 is exposed on the host machine port 80.
- expose : Declares the ports to expose.
- networks : Associates the service with one or more Docker networks. In this example, internalnet is defined as the network this service belongs to.
- internalnet : the name of the Docker network. here it is defined as internalnet.
- driver: bridge : specifies the type of network driver to use. bridge is the default driver for Docker networks and it allows containers to communicate with each other on the same host.

III.3.5 Running container



edtc		Running (2/2)	0%	
cvtlying-fnd-container 12d6d6fc30ae	edtc-cvtlyin	Running	0%	80:80
crazy_booth e7f10eccc8bc	edtc-cvtlyin	Running	0%	

Figure III.13: Running container