



Université Abdelmalek Essaadi
Faculté des Sciences et techniques de Tanger
Département Génie Informatique
Année universitaire : 2021/2022
Cycle d'ingénieur : LSI, S2



logiciels et systèmes intelligent

Projet de fin de module

développement Front end

Réalisation d'une progressive web application

clone de Fiverr

Réaliser par :

DAAOUAN MOHAMMED

FRIKH SAID

Encadré par :

Pr. AACHAK LOTFI

Année universitaire 2022-2023



REMERCIEMENTS

Nous adressons nos sincères remerciements à notre Professeur **Pr. Lotfi EL AACHAK** qui mérite toute notre gratitude. Merci pour sa bonne volonté, sa patience et ses précieux conseils, ainsi que pour la pertinence de ses remarques.

Ces remerciements vont tout au corps professoral et administratif de la **Faculté des Sciences et Techniques de Tanger**, spécialement ceux du département Informatique pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée.

TABLE DES MATIÈRES

Table des figures	v
Liste des tableaux	vi
Introduction	1
1 Analyse et Conception	2
1.1 Introduction	2
1.2 Identifications des acteurs	2
1.2.1 Un client	2
1.2.2 Un freelanceur	2
1.2.3 Un visiteur	3
1.3 Diagramme de cas d'utilisation	3
1.4 Diagramme d'activité	4
1.5 Diagramme de classes	5
1.6 Conclusion	6
2 Introduction aux technologies utilisées	7
2.1 Introduction	7
2.2 Technologies du Front-End	7
2.2.1 Angular	7
2.2.2 Angular Material	8
2.2.3 Lazy Loading	8
2.2.4 modal forms	8
2.2.5 guards	8
2.3 Technologies du Back-End	9
2.3.1 Node.js	9
2.3.2 Express.js	9
2.3.3 GraphQL	10
2.3.4 Apollo server	10
2.4 Base de données	11
2.4.1 MongoDB	11
2.4.2 Mongoose	11

2.5 Système de contrôle de version	12
2.5.1 Git	12
3 Présentation de l'application web	13
3.1 Introduction	13
3.2 First page	13
3.3 Home page	14
3.4 register form	15
3.5 login form	16
3.6 Gig page	17
3.7 Creat Gig form	18
3.8 payment form	19
Conclusion et perspectives	20
Bibliographie	21

TABLE DES FIGURES

1.1	Diagramme de cas d'utilisation	3
1.2	Diagramme d'activité	4
1.3	Diagramme de classes	5
2.1	Angular	7
2.2	Angular material	8
2.3	Nodejs	9
2.4	Expressjs	9
2.5	GraphQL	10
2.6	Apollo server	10
2.7	MongoDB	11
2.8	Mongoose	11
2.9	Git	12
3.1	First page 1	13
3.2	First page 2	14
3.3	Home page	14
3.4	Register form	15
3.5	Login form	16
3.6	Gig page 1	17
3.7	Gig page 2	17
3.8	Creat Gig form	18
3.9	payment form	19

LISTE DES TABLEAUX

INTRODUCTION

Le développement Front-End occupe une place primordiale dans la création d'applications web modernes et conviviales. Dans le cadre de notre projet de fin de module, notre objectif est de mettre en place une application web en utilisant le concept de Web Progressive Application (PWA). Cette application sera conçue comme un clone de plateformes bien connues telles que Fiverr ou Freelancer, qui facilitent les interactions entre freelancers et clients à la recherche de services spécifiques.

L'objectif de notre projet est de créer une application web qui offre une expérience utilisateur fluide, rapide et réactive, à la fois sur ordinateur et sur les appareils mobiles. En utilisant les principes de la PWA, nous assurerons que l'application est accessible même en l'absence d'une connexion Internet stable, permettant ainsi aux utilisateurs de l'utiliser en mode hors ligne.

Notre objectif est de fournir aux utilisateurs une expérience similaire à celle des plateformes existantes, avec des fonctionnalités telles que la création de profils d'utilisateur, la recherche et la publication de projets, la gestion des transactions et des paiements, et bien plus encore. Nous accorderons une attention particulière à l'optimisation des performances et à l'accessibilité de l'application, afin de garantir une expérience utilisateur optimale pour tous les utilisateurs.

— CHAPITRE 1 —

ANALYSE ET CONCEPTION

1.1 Introduction

Dans ce chapitre, nous allons aborder la conception de notre application web qui vise à créer une plate-forme similaire à Fiverr, Freelancer, etc. Nous examinerons les différents aspects de la conception, y compris les acteurs impliqués, les cas d'utilisation de chaque acteur, les diagrammes d'activité et les classes utilisées dans l'application.

1.2 Identifications des acteurs

Nous commencerons par identifier les acteurs principaux de notre application web. Ces acteurs jouent des rôles clés dans l'utilisation et l'interaction avec notre plate-forme. Nous détaillerons les caractéristiques et les responsabilités de chaque acteur, en mettant l'accent sur leur rôle dans le processus global. Dans notre application web, nous identifions trois acteurs principaux : le client, le freelanceur et le visiteur

1.2.1 Un client

Le client est l'acteur principal de notre application, tout comme sur Fiverr. Il s'agit de l'utilisateur qui recherche des services spécifiques pour répondre à ses besoins. Le client peut créer un compte, parcourir les services proposés par les freelancers, passer des commandes, interagir avec les freelancers, négocier les prix et effectuer des paiements.

1.2.2 Un freelanceur

Le freelanceur est un professionnel indépendant qui propose ses services sur notre plate-forme, tout comme sur Fiverr. Le freelanceur peut créer un profil, détailler ses compétences, proposer des services, recevoir des commandes de la part des clients, communiquer avec les clients, établir des devis et recevoir des paiements pour ses services rendus.

1.2.3 Un visiteur

Le visiteur est un utilisateur non enregistré qui accède à notre application sans créer de compte, tout comme sur Fiverr. Les visiteurs peuvent explorer les services proposés par les freelancers et avoir un aperçu des fonctionnalités de notre application. Cependant, pour bénéficier de toutes les fonctionnalités et interagir avec les clients et les freelancers, un visiteur doit créer un compte en tant que client ou freelanceur.

1.3 Diagramme de cas d'utilisation

Les cas d'utilisation décrivent les différentes actions et interactions qu'un acteur peut effectuer dans notre application web.

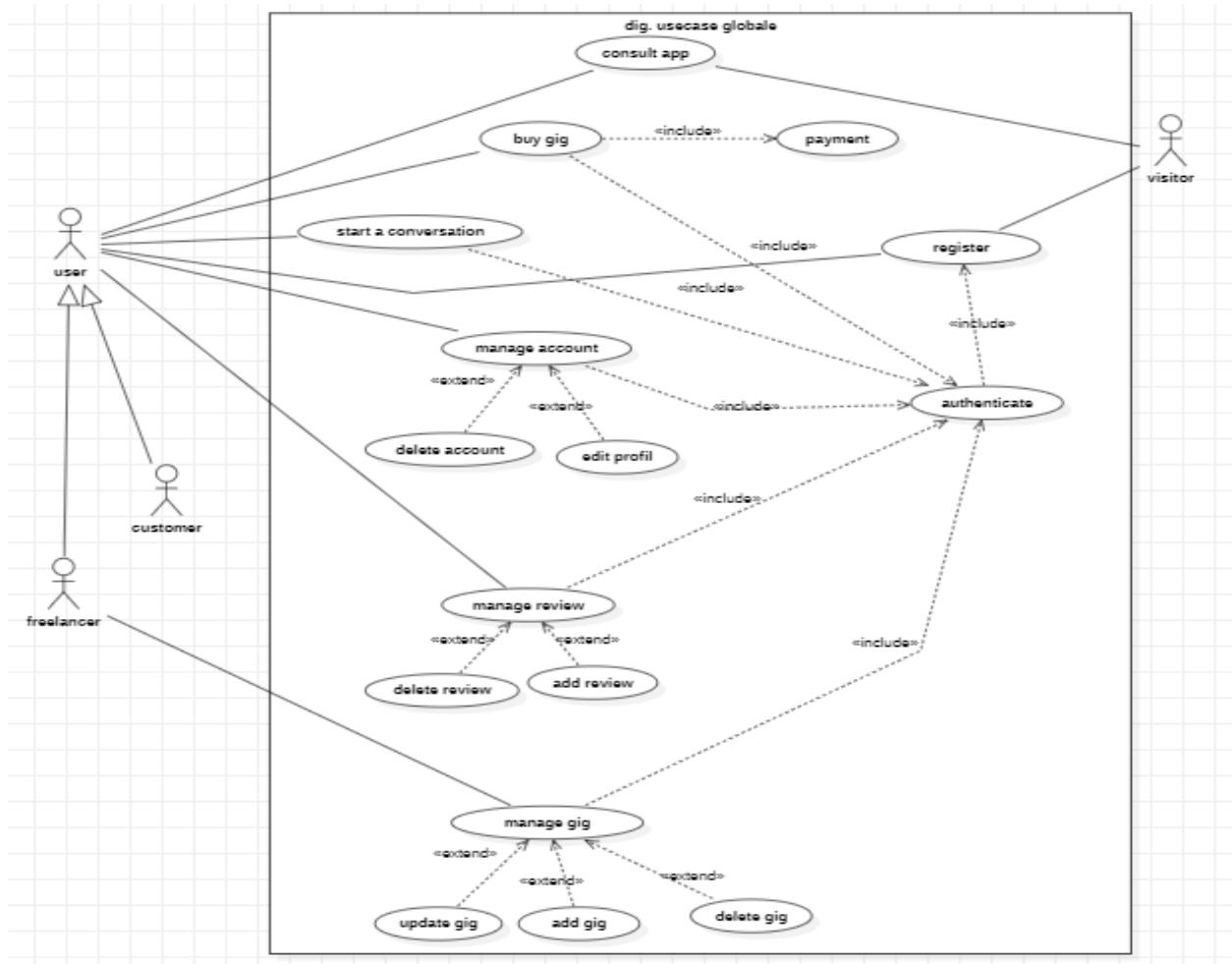


FIGURE 1.1 – Diagramme de cas d'utilisation

1.4 Diagramme d'activité

Les diagrammes d'activité nous permettent de visualiser le flux de travail et les étapes impliquées dans différents processus de l'application. Nous créerons des diagrammes d'activité pour représenter les actions effectuées par les acteurs et les différentes étapes du processus, du début à la fin. Cela nous aidera à comprendre la logique de fonctionnement de notre application web.

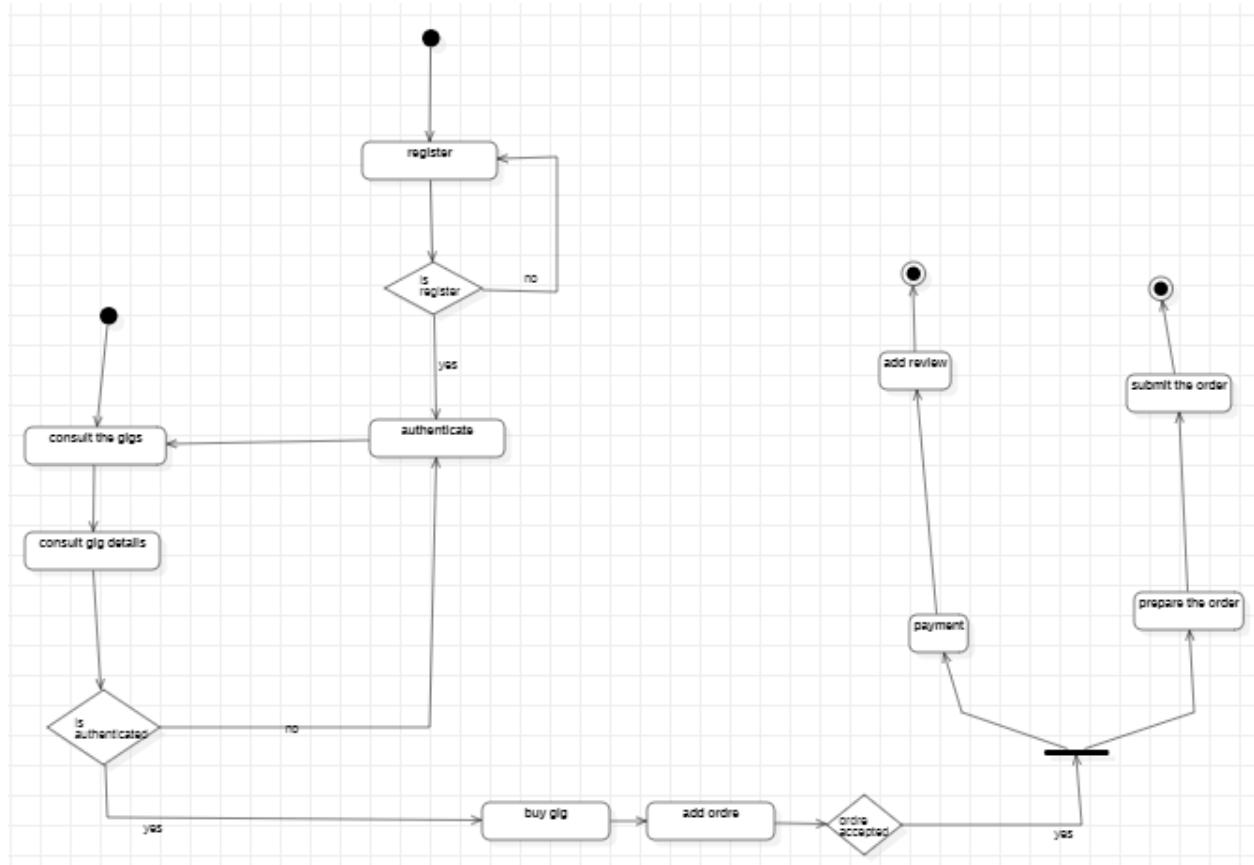


FIGURE 1.2 – Diagramme d'activité

1.5 Diagramme de classes

Enfin, nous examinerons les classes utilisées dans notre application web. Nous identifierons les entités principales et les objets clés qui interagissent dans notre système. Nous définirons les attributs et les méthodes de chaque classe, en mettant en évidence leurs relations et leur rôle dans le fonctionnement global de l'application.

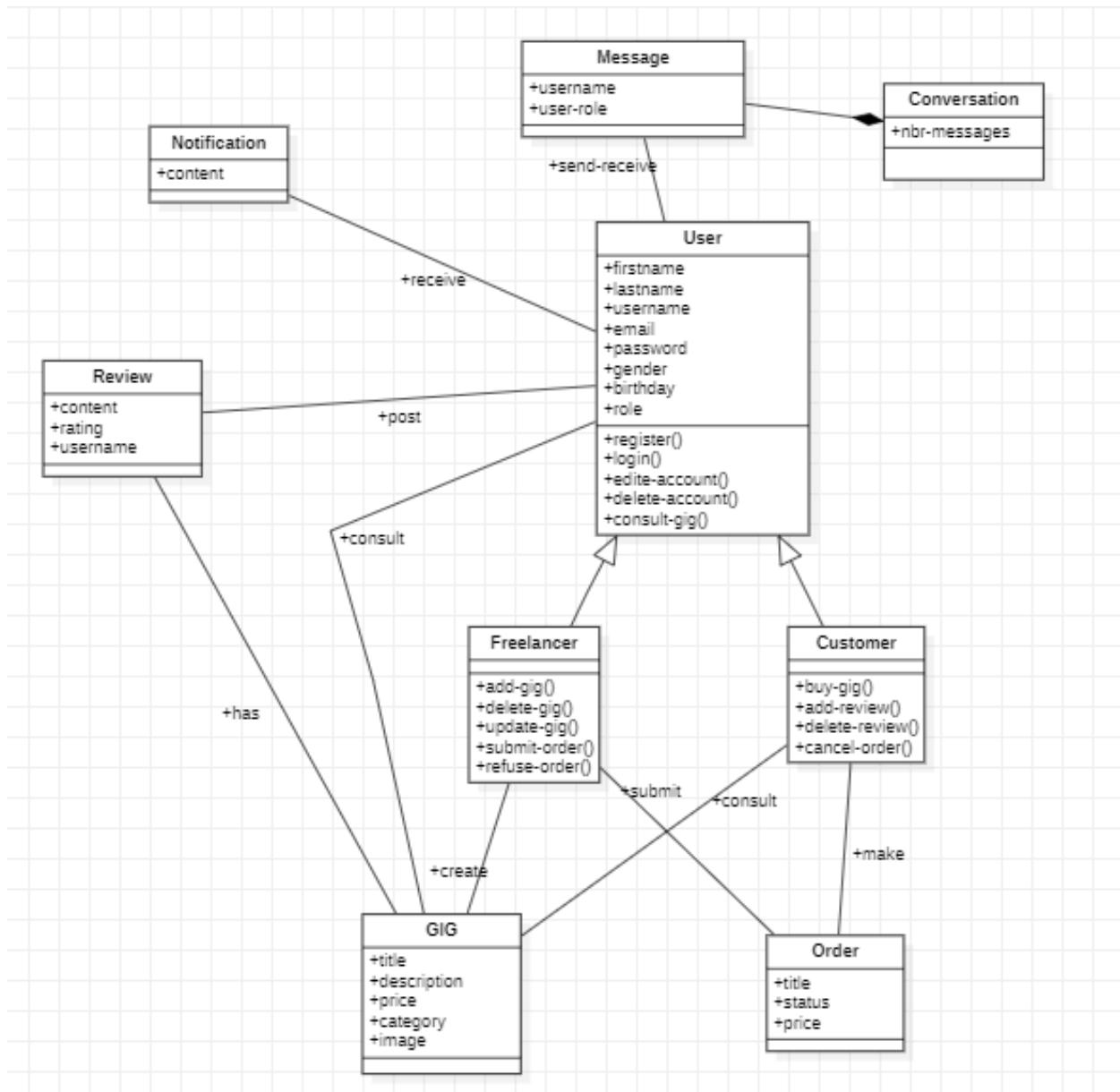


FIGURE 1.3 – Diagramme de classes

1.6 Conclusion

En résumé, ce chapitre se concentre sur la conception de notre application web clone de Fiverr, Freelancer, etc. Nous explorerons les acteurs impliqués, les cas d'utilisation de chaque acteur, les diagrammes d'activité et les classes utilisées. Cette analyse approfondie jettera les bases solides de notre projet et nous permettra de passer à la prochaine phase de développement avec une vision claire et une compréhension approfondie de notre application.

— CHAPITRE 2 —

INTRODUCTION AUX TECHNOLOGIES UTILISÉES

2.1 Introduction

Ce chapitre est dédié à la présentation des différentes technologies que nous avons choisies pour développer notre application web. Nous explorerons à la fois le côté Front-End et le côté Back-End, en mettant l'accent sur les outils et les frameworks qui nous permettront de créer une expérience utilisateur robuste et une architecture solide.

2.2 Technologies du Front-End

2.2.1 Angular

Angular est un framework de développement d'applications web développé par Google. Il offre une structure solide pour la création d'applications web évolutives et réactives. Avec Angular, nous pouvons organiser notre code de manière modulaire, gérer les routes, gérer les interactions utilisateur et manipuler efficacement les données.

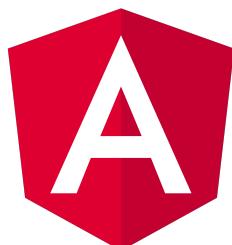


FIGURE 2.1 – Angular

2.2.2 Angular Material

Angular Material est une bibliothèque de composants d'interface utilisateur basée sur les principes du Material Design. Elle offre une vaste gamme de composants prêts à l'emploi tels que des boutons, des formulaires, des tables, des menus, des dialogues, etc. Angular Material facilite la création d'interfaces utilisateur modernes, cohérentes et esthétiquement plaisantes.

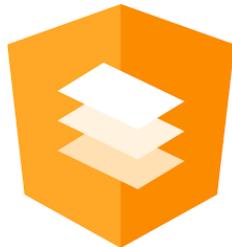


FIGURE 2.2 – Angular material

2.2.3 Lazy Loading

Le chargement paresseux (Lazy Loading) est une technique qui permet de charger les modules et les composants d'une application Angular de manière dynamique, uniquement lorsque cela est nécessaire. Cela permet de réduire le temps de chargement initial de l'application et d'améliorer les performances globales. Angular offre une prise en charge native du chargement paresseux, ce qui facilite l'optimisation de notre application en termes de performances.

2.2.4 modal forms

Les modal forms, ou formulaires modaux, sont des composants qui s'affichent dans une fenêtre contextuelle (modal) et permettent à l'utilisateur de saisir des informations ou d'effectuer des actions spécifiques. Angular facilite la création de modal forms en utilisant des composants tels que MatDialog, qui nous permettent de créer des fenêtres modales interactives pour afficher des formulaires ou des messages contextuels.

2.2.5 guards

Les guards, ou gardiens, sont des mécanismes utilisés dans Angular pour contrôler l'accès aux routes de l'application. Les guards nous permettent de définir des règles et des conditions pour autoriser ou refuser l'accès à certaines parties de notre application en fonction des autorisations de l'utilisateur. Ils sont utiles pour sécuriser les pages, les fonctionnalités ou les ressources sensibles de notre application.

2.3 Technologies du Back-End

2.3.1 Node.js

Node.js est un environnement d'exécution JavaScript côté serveur basé sur le moteur JavaScript V8 de Google. Il permet d'exécuter du code JavaScript en dehors d'un navigateur, ce qui ouvre la porte à de nombreuses possibilités pour le développement web. Node.js offre une architecture orientée événements et une manipulation asynchrone des entrées/sorties, ce qui le rend idéal pour les applications en temps réel et les tâches gourmandes en ressources. Il est également doté d'une vaste bibliothèque de modules prêts à l'emploi, ce qui facilite le développement rapide et efficace d'applications web.



FIGURE 2.3 – Nodejs

2.3.2 Express.js

Express.js est un framework web minimaliste et flexible pour Node.js. Il offre une couche d'abstraction légère au-dessus de Node.js, facilitant la création de serveurs web et d'APIs RESTful. Express.js permet de définir facilement des routes, de gérer les requêtes et les réponses, et d'intégrer des middlewares pour la gestion des sessions, la validation des données, l'authentification, etc. Avec Express.js, le développement d'applications web devient plus structuré, maintenable et évolutif.



FIGURE 2.4 – Expressjs

2.3.3 GraphQL

GraphQL est un langage de requête et une spécification pour les APIs (Application Programming Interfaces). Contrairement aux architectures REST traditionnelles, GraphQL permet aux clients de spécifier exactement les données dont ils ont besoin, évitant ainsi le problème du surchargement ou du sous approvisionnement de données. Avec GraphQL, les clients peuvent interroger plusieurs ressources en une seule requête et obtenir les données exactes demandées, ce qui améliore les performances et l'efficacité des applications web.

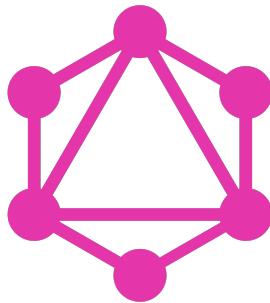


FIGURE 2.5 – GraphQL

2.3.4 Apollo server

Apollo Server est une plateforme puissante pour créer des APIs GraphQL. Elle simplifie la configuration et la mise en place du serveur, s'intègre avec des frameworks populaires comme Node.js et Express.js, et propose des fonctionnalités avancées de gestion des données. Avec des capacités intégrées de sécurité et d'authentification, elle garantit une API sécurisée. Apollo Server offre également des outils de surveillance et d'analyse via Apollo Studio. Globalement, Apollo Server est une solution complète pour construire des APIs GraphQL robustes et évolutives.

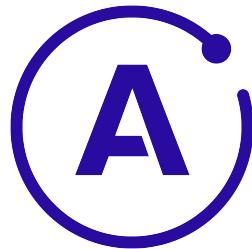


FIGURE 2.6 – Apollo server

2.4 Base de données

2.4.1 MongoDB

MongoDB est une base de données NoSQL orientée document. Elle offre une grande flexibilité dans la gestion des données, permettant de stocker des documents JSON de manière native. MongoDB est conçu pour être évolutif et performant, adapté aux applications web modernes qui nécessitent une gestion efficace des données non structurées ou semi-structurées. Avec MongoDB, nous pouvons stocker, interroger et manipuler facilement nos données de manière flexible.



FIGURE 2.7 – MongoDB

2.4.2 Mongoose

Mongoose est une bibliothèque JavaScript qui facilite l’interaction avec la base de données MongoDB. Il fournit une couche d’abstraction supplémentaire qui simplifie la création de schémas, la validation des données, les requêtes et les opérations de base de données. Mongoose permet de définir des modèles de données clairs et cohérents, facilitant ainsi le développement et la maintenance de notre application back-end.



FIGURE 2.8 – Mongoose

2.5 Système de contrôle de version

2.5.1 Git

Git est un système de contrôle de version distribué gratuit et open source conçu pour tout gérer, des petits aux très grands projets, avec rapidité et efficacité.



FIGURE 2.9 – Git

— CHAPITRE 3 —

PRÉSENTATION DE L'APPLICATION WEB

3.1 Introduction

Cette section comporte des captures d'écran de quelques interfaces de l'application réalisée .

3.2 First page

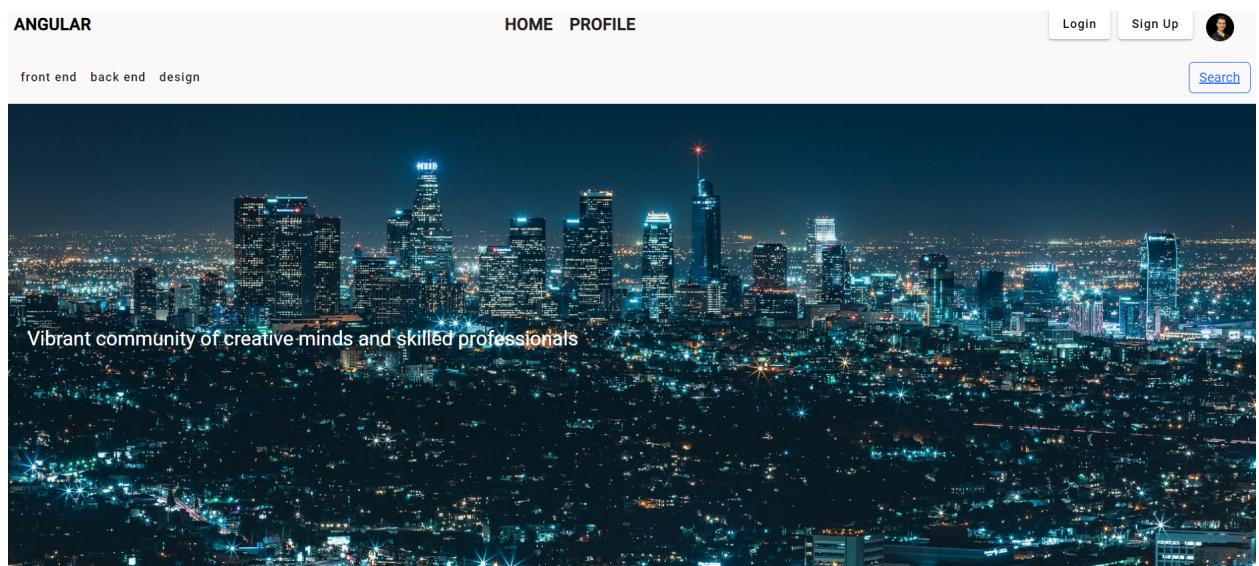


FIGURE 3.1 – First page 1

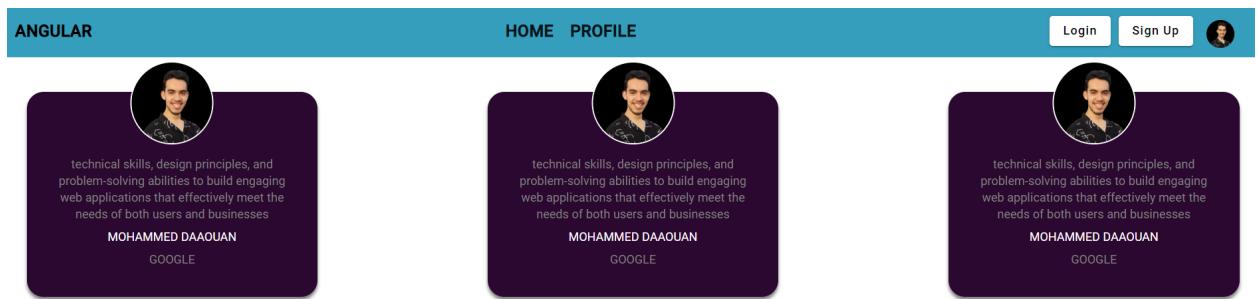


FIGURE 3.2 – First page 2

3.3 Home page

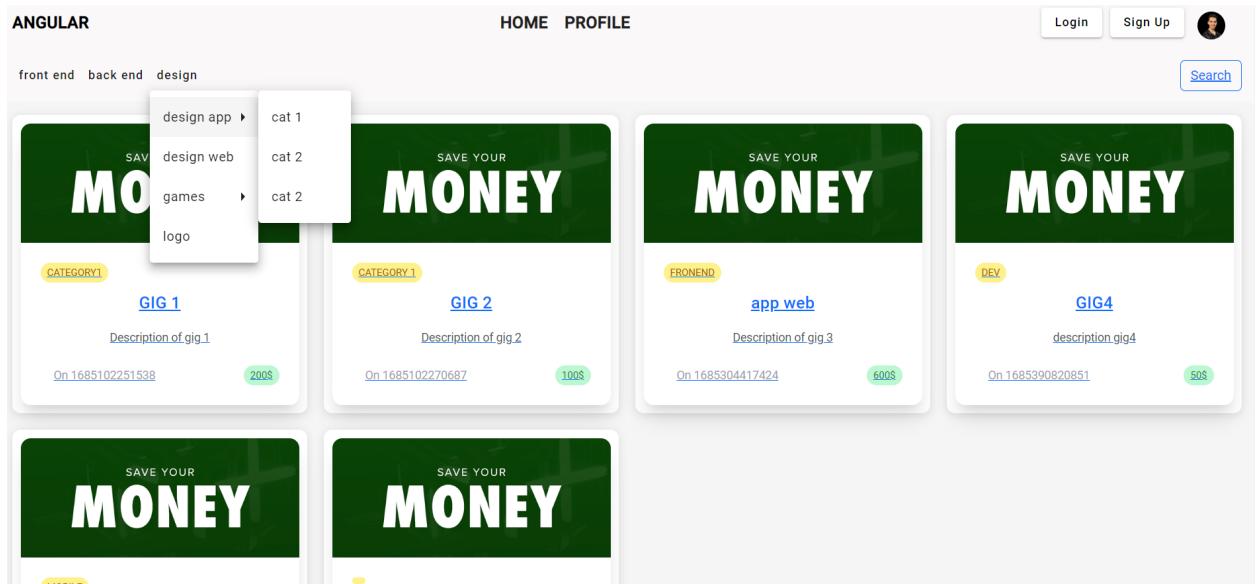


FIGURE 3.3 – Home page

3.4 register form

1 Fill Basic ... — 2 Fill out conta... — 3 Fill out ad...

First Name*

Last Name*

Email*

Birthday*

Gender

Male Female

Next

FIGURE 3.4 – Register form

3.5 login form

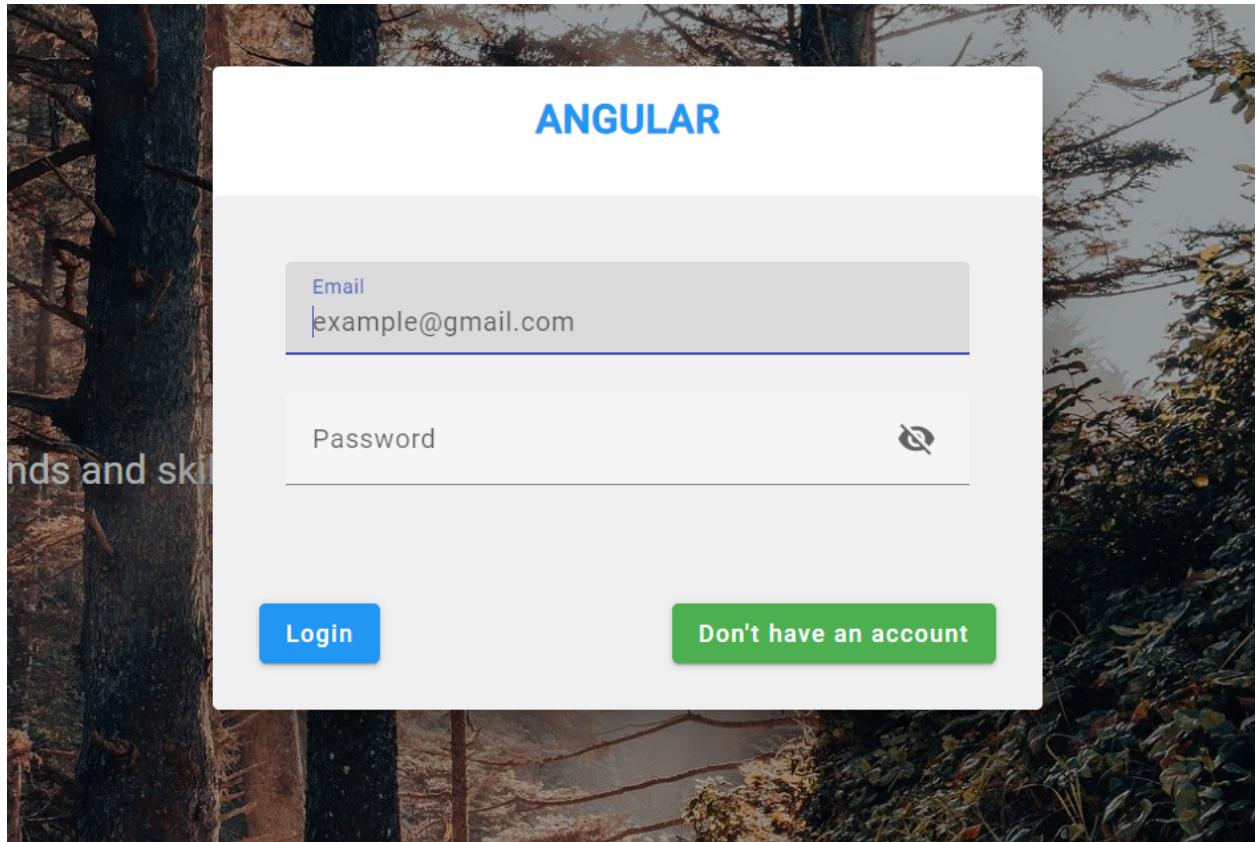


FIGURE 3.5 – Login form

3.6 Gig page

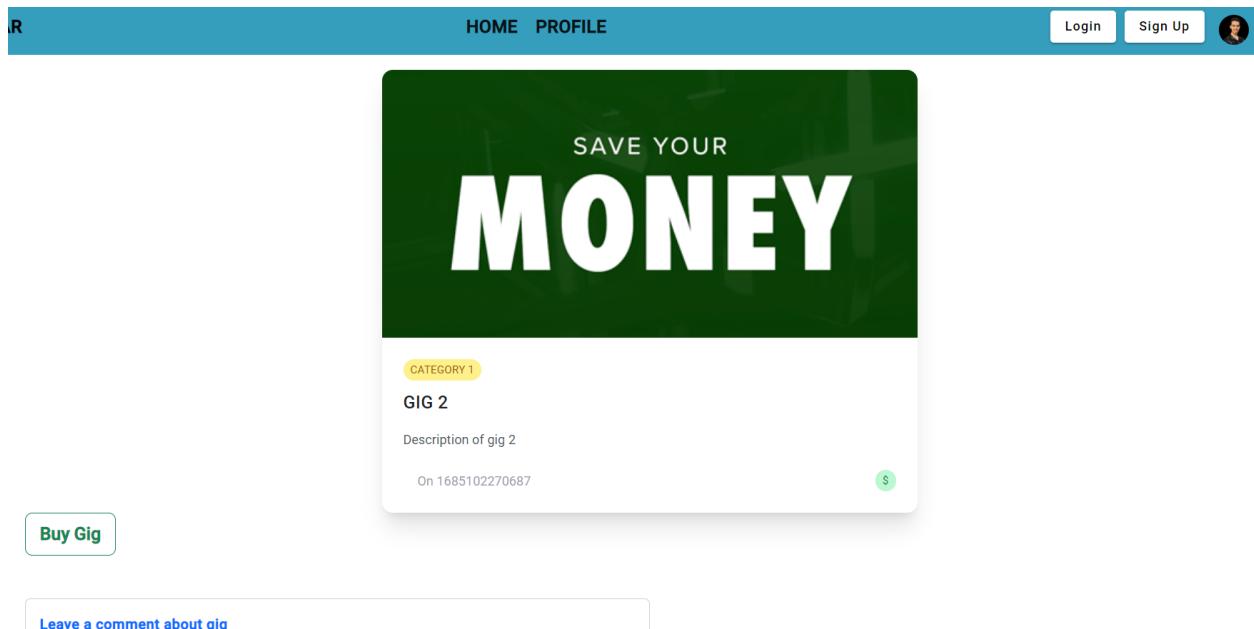


FIGURE 3.6 – Gig page 1

The screenshot displays a comment section and a list of comments for a gig. The comment section has a heading 'Leave a comment about gig' and a text input field asking 'Tell us how you found this gig'. It includes a 'Comment' label and a 'Add your comment here' input field with a 'Add comment' button. Below this is a 'Comments (15,256)' section. The first comment is by 'said frikh' (Last update - MAR 20, 2023) with the text: 'Lorem ipsum dolor sit amet consectetur adipisicing elit. Quasi debitis voluptatibus ipsa provident. Quis corrupti illo, deleniti odio magni dignissimos culpa ipsam sit fugit sunt ducimus delectus itaque asperiores placeat!'. There are 'Reply' and 'View Reply' links below the comment. Another comment by 'said frikh' (Last update - MAR 20, 2023) is partially visible at the bottom.

FIGURE 3.7 – Gig page 2

3.7 Creat Gig form

The screenshot shows a modal window titled "Add new gig". The form contains four input fields: "title" (with placeholder "title" and a blue cursor), "Category" (a dropdown menu with a downward arrow icon), "description" (with placeholder "description"), and "price" (with placeholder "price"). Below these fields is a "Image" section with a "Choisir un fichier" button and a "Aucun fichier choisi" message. At the bottom right is a green button labeled "New Article" with a white arrow icon.

FIGURE 3.8 – Creat Gig form

3.8 payment form

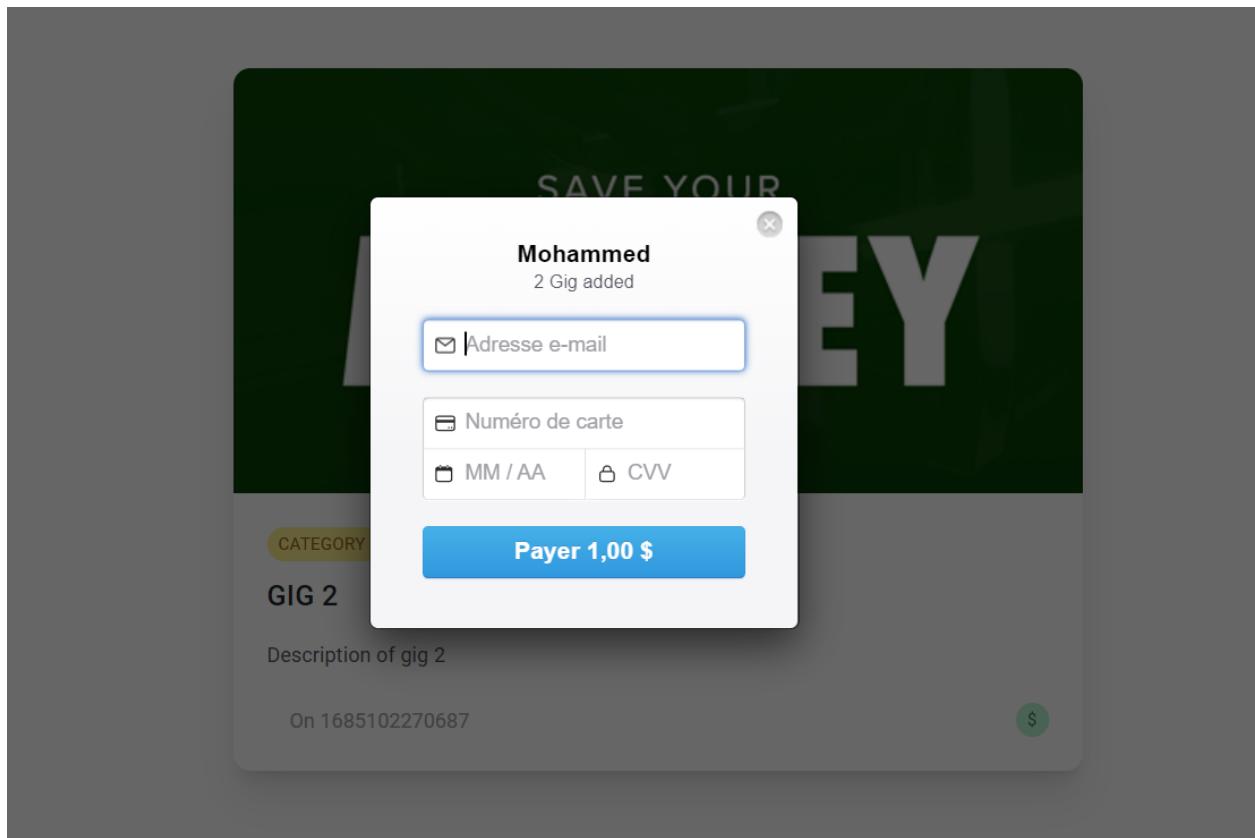


FIGURE 3.9 – payment form

CONCLUSION ET PERSPECTIVES

Perspective

BIBLIOGRAPHIE

[1]