



Fleurtissimo

Robin Simonneau

Bilel Taieb

Rujdy Hasni

<https://github.com/RoroPC/dev-web-cytech>

2023-2024

Table des matières

<i>Introduction</i>	3
<i>I. Présentation</i>	4
1. Le Frontend	4
2. Le Backend.....	4
<i>II. Outils d'organisations</i>	5
<i>III. Difficultés :</i>	6
1. CORS	6
2. CSRF.....	7
<i>IV. Utilisation</i>	8
1. Docker	8
2. Sans docker	8
3. Problèmes.....	9
<i>Conclusion</i>	10
<i>V. Annexes</i>	11
1. Diagramme UML.....	11
Diagramme de classe	11
Diagrammes de cas d'utilisations.....	12
2. Image de présentation de l'application.....	14
Page d'accueil.....	14
Page d'un produit	15
Page de demande de contact	16
Page du panier.....	17
Page de connexion.....	18
Page d'inscription.....	19

Introduction

Au cours de notre formation d'ingénieur à CY Tech, nous avons dû concevoir et réaliser une application web de vente de fleurs que nous avons nommée Fleurtissimo. Ce travail nous a permis d'étoffer nos compétences dans la conception d'une base de données, mais aussi dans la conception d'une application, tout en nous permettant d'acquérir de nouvelles compétences dans le développement web.

Dans ce rapport, nous allons commencer par décrire les outils utilisés pour réaliser ce projet, puis nous parlerons de l'organisation mise en place. Enfin, nous aborderons les quelques difficultés que nous avons rencontrées lors de la création de cette application. Dans les deux dernières parties de ce rapport, nous expliquerons comment mettre en place l'application et nous afficherons également les diagrammes UML qui nous ont servis pour le développement de ce projet.

I. Présentation

Tout d'abord, nous allons commencer par présenter l'application et les technologies utilisées pour la concevoir.

Fleurtissimo est un site de vente de fleurs en ligne, subdivisé en deux parties majeures :

1. Le Frontend

Le frontend est codé grâce au *framework React*. React est une bibliothèque JavaScript (ou TypeScript) créée en 2013 et maintenue par Meta. Elle permet de créer des applications web monopages (SPA - Single Page Applications), c'est-à-dire des applications web ne contenant qu'une seule page. React divise l'application en composants qu'elle peut afficher ou modifier de façon dynamique en fonction des données, ce qui offre une expérience utilisateur fluide et réactive.

2. Le Backend

Le backend est géré par Django et son extension Django REST Framework. Django est un *framework Python* créé en 2003 qui permet de développer des applications web robustes. Django REST Framework est une extension de Django qui permet de l'utiliser comme une API REST. Une API (Application Programming Interface) est une interface de programmation qui met à disposition des données pour qu'elles soient utilisées par d'autres applications, comme notre application React.

Les avantages d'utiliser une API par rapport à une application web classique sont nombreux. Premièrement, cela permet d'utiliser un *framework frontend* comme React, qui donne une impression de fluidité et de réactivité à l'utilisateur final. De plus, cela permet de réutiliser les données dans d'autres applications, comme une application mobile par exemple.

Pour résumer, React communique avec l'API Django qui se charge de modifier la base de données.

A noté que le site est chiffré en HTTPS afin de protéger les données qui transite entre le *frontend* et le *backend*.

II. Outils d'organisations

Afin de bien nous organiser et de tenir compte des choses qu'il nous restait à faire, nous avons décidé d'utiliser une méthode de gestion de projet très connue, la méthode Kanban. Cette méthode consiste à séparer les tâches en tickets et à les placer dans quatre catégories selon leur avancement : **À faire**, **En cours**, **Bloqué** ou **Terminé**. Cette méthode nous permet d'affecter des « tickets » à des membres du groupe et donc de voir à tout moment qui s'occupe de quelle tâche et quelles sont les tâches qu'il reste à faire.

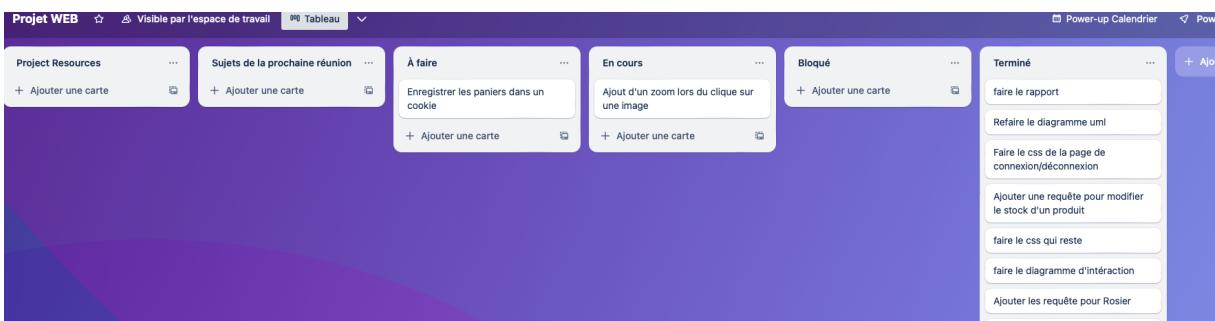


FIGURE 1: CAPTURE D'ECRAN DE NOTRE TRELLLO

Pour mettre en place ce système, nous avons décidé d'utiliser un outil très connu : Trello. Trello est un outil de gestion de projets qui utilise cette méthode. Il existe plein d'alternatives, mais Trello était celle que nous maîtrisions le mieux.

En plus de la gestion de projet, il est important de faire de la gestion de code afin de tenir compte de qui a fait des changements et donc éviter d'empiéter sur le travail des autres. Pour ce faire, nous avons décidé d'utiliser l'outil le plus connu pour cela : Git. Git permet de faire de la gestion de versions de code et nous a permis de mieux nous organiser dans la partie réalisation du projet notamment. En plus de Git, nous avons utilisé GitHub, qui est un outil permettant entre autres de conserver son code source et le versionnage de son projet. Avec ces trois outils, nous avons pu mettre en place une gestion de projet efficace.

III. Difficultés :

Durant le développement de ce projet nous avons pu rencontrer différentes difficultés qui n'était pas prévu à la base, cette partie du rapport va nous permettre de les expliqués, et d'expliquer les moyens mis en place pour les surmontés.

1. CORS

Le CORS ou **Cross-Origin Resource Sharing** est un mécanisme mis en place dans les navigateurs web récents. Ce mécanisme régit la façon dont le navigateur doit interagir avec les requêtes vers d'autres domaines non prévus.

Pour faire simple, le navigateur bloque par défaut les requêtes envoyées vers un autre domaine que le domaine actuellement consulté. Ce blocage évite qu'un potentiel attaquant puisse récupérer des données de connexion d'un utilisateur en le redirigeant vers son propre domaine.

Cependant, dans certains cas, il se peut que le serveur ait besoin de récupérer des informations sur une source externe. Par exemple, pour charger une vidéo YouTube dans un site web, le navigateur doit faire une requête vers le domaine youtube.com. Dans ce cas-là, le serveur de l'application web couramment utilisé doit approuver les requêtes vers ce domaine.

Une des difficultés que nous avons eues est due au blocage du CORS par le navigateur. En effet, cela est dû à la façon dont fonctionne notre application en développement. React doit utiliser un serveur web pour fonctionner. En développement local, l'adresse web utilisée par React est `http://localhost:3000`, or notre serveur Django qui fait tourner l'API utilise l'adresse `http://127.0.0.1:8000`. Bien que `localhost` et `127.0.0.1` représentent tous deux l'adresse IP locale de l'ordinateur de développement, les domaines diffèrent, ce qui provoquait ce blocage par le CORS.

Pour régler ce problème, nous avons dû autoriser sur le serveur Django la possibilité de faire des requêtes à la fois vers `localhost` et vers `127.0.0.1`.

2. CSRF

La deuxième grosse difficulté rencontrée était due aux mesures de sécurité mises en place pour éviter les *CSRF*.

CSRF ou **Cross-Site Request Forgery** est une vulnérabilité connue en développement web. Cette attaque peut notamment permettre de se connecter avec le compte d'un autre utilisateur.

En effet, dû à l'utilisation d'une architecture utilisant une API, lorsque l'utilisateur se connecte avec son nom d'utilisateur et son mot de passe, le serveur génère un cookie de session. Ce cookie prouve que l'utilisateur est bien celui qu'il prétend être. Cependant, ce cookie étant transporté dans des requêtes HTTP, il peut être intercepté par un tiers. Ce tiers peut ensuite l'utiliser pour se connecter sur le compte de l'utilisateur.

Étant donné cette vulnérabilité, les navigateurs ont mis en place des sécurités pour protéger ces cookies, la plus simple étant d'utiliser des requêtes HTTPS qui sont chiffrées contrairement aux requêtes HTTP. De ce fait, le navigateur bloque toute requête HTTP qui comporte des cookies de session.

Pour régler ce problème, il a fallu chiffrer nos requêtes vers l'API. Cependant, ce chiffrement utilise normalement des certificats d'authenticité SSL qui sont générés par un service tiers de confiance comme Let's Encrypt par exemple. Or, notre application étant encore en développement, et donc non accessible depuis l'extérieur, nous avons dû utiliser un outil pour créer nous-mêmes des certificats d'authenticité auto-signés, mais qui permettent le chiffrement de nos requêtes. Cependant, cette solution est temporaire et fonctionnelle que sur le serveur de développement. En effet, pour éviter l'utilisation de certificats auto-signés, le navigateur prévient l'utilisateur que l'application qu'il utilise n'est pas sécurisée, ce qui n'est pas optimal pour démontrer le sérieux d'une application. Si l'application passait en environnement de production, il faudrait alors faire une demande à un partenaire de confiance afin d'obtenir un certificat SSL valide.

IV. Utilisation

Dans cette partie nous allons vous expliquer comment récupérer l'application et l'exécuter sur une machine.

Afin de pouvoir utiliser le site en local il existe deux méthodes :

Répertoire Github : <https://github.com/RoroPC/dev-web-cytech>

1. Docker

La méthode la plus simple, qui ne nécessite aucune configuration particulière, est d'utiliser Docker. Pour cela, il suffit de récupérer le répertoire du projet sur GitHub, d'avoir Docker installé, puis d'exécuter la commande :

`docker-compose up`

Cela devrait lancer l'API et le *frontend*.

Le *frontend* sera accessible ici : <http://localhost:3000/>

Le panel administrateur sera disponible ici : <https://127.0.0.1:8000>

L'email de l'administrateur est : *jf@gmail.com*

Le mot de passe est : *testtest*

Le panel administrateur permet d'accéder aux données du site comme : Les utilisateurs, les commandes, les fleurs et leurs catégories.

2. Sans docker

Pour lancer sans docker, il faudra : avoir python en version 3.12, et la version 10.8.0 de *npm* installer.

1. Lancer le *frontend*

Pour lancer le *frontend*, il faudra exécuter ces commandes :

`cd front`

`npm install`

`npm run dev`

2. Lancer le *backend*

Pour lancer le *backend* il faudra exécuter ces commandes (sur un environnement UNIX):

`cd back`

`python3 -m venv .venv`

`source .venv/bin/activate`

```
pip install -r requirements.txt
```

```
python3 manage.py runserver_plus --cert-file cert.pem --key-file key.pem
```

3. Problèmes

Si la connexion et ou l'inscription ne fonctionne pas, c'est sûrement à cause des protections du navigateur qui bloque les requêtes sur les certificats auto-signés.

Si dans la console du navigateur un message du type : **NET::ERR_CERT_AUTHORITY_INVALID**

Il faut désactiver la protection de chrome, pour cela il faut (pour chrome):

- Aller ici chrome://flags/#allow-insecure-localhost
- Mettre ce paramètre : **Allow invalid certificates for resources loaded from localhost** à **enabled**

Conclusion

Pour conclure, on peut dire que ce projet nous a permis d'approfondir nos connaissances dans des sujets tels que la gestion de projets avec Trello, la conception d'une application web allant de la création de la base de données jusqu'à la création de l'architecture logicielle avec l'utilisation de diagrammes de classes ainsi que de diagrammes de cas d'utilisation. Nous avons également acquis des compétences dans la création d'applications web avec React et Django, tout en approfondissant nos connaissances de Docker.

V. Annexes

1. Diagramme UML

Diagramme de classe

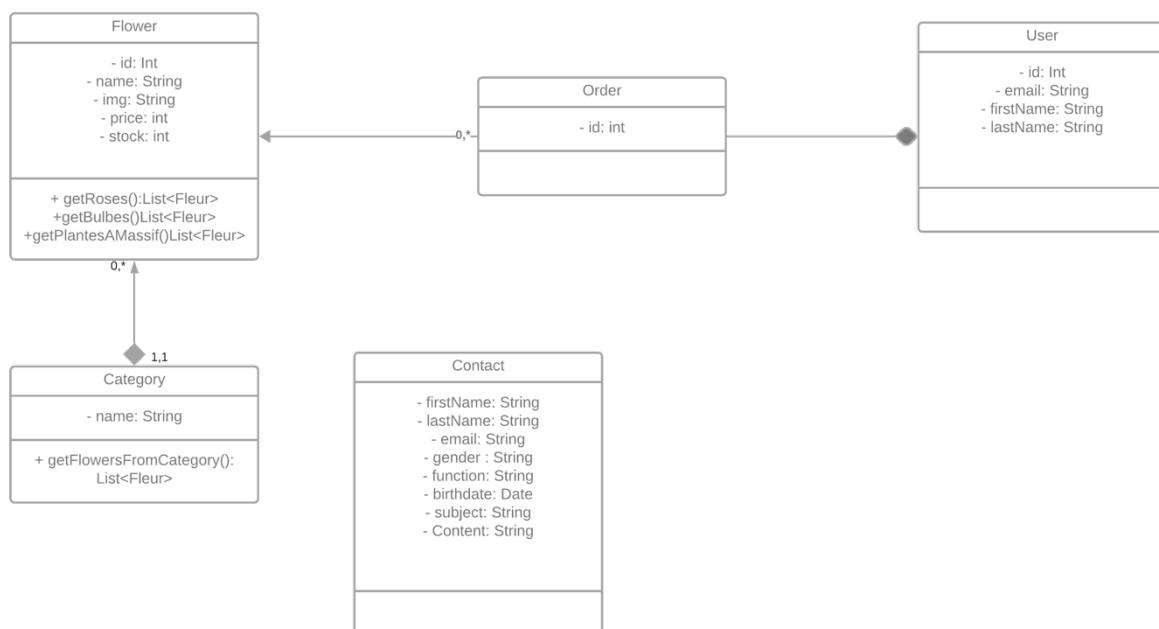


FIGURE 2: DIAGRAMME DE CLASSE DE L'APPLICATION

Diagrammes de cas d'utilisations

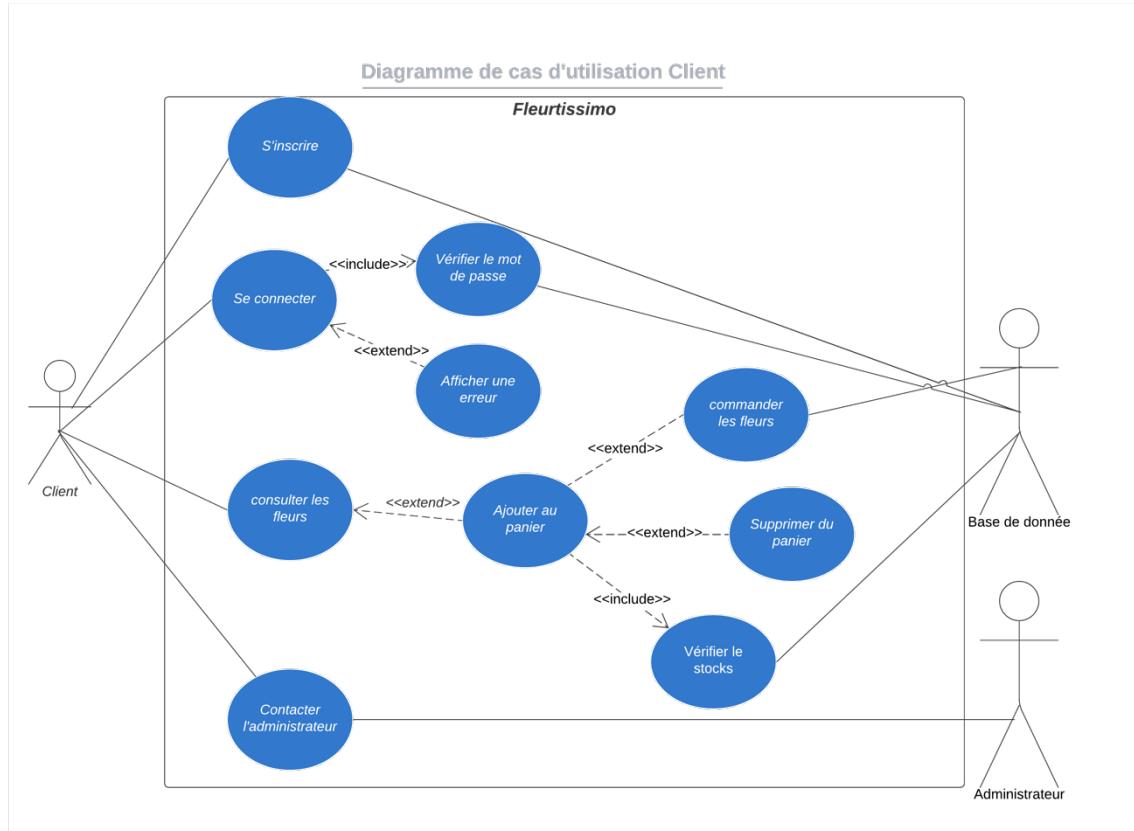


FIGURE 3: CAS D'UTILISATION MONTRANT LES INTERACTIONS DE L'UTILISATEUR

Diagramme de cas d'utilisation Administrateur

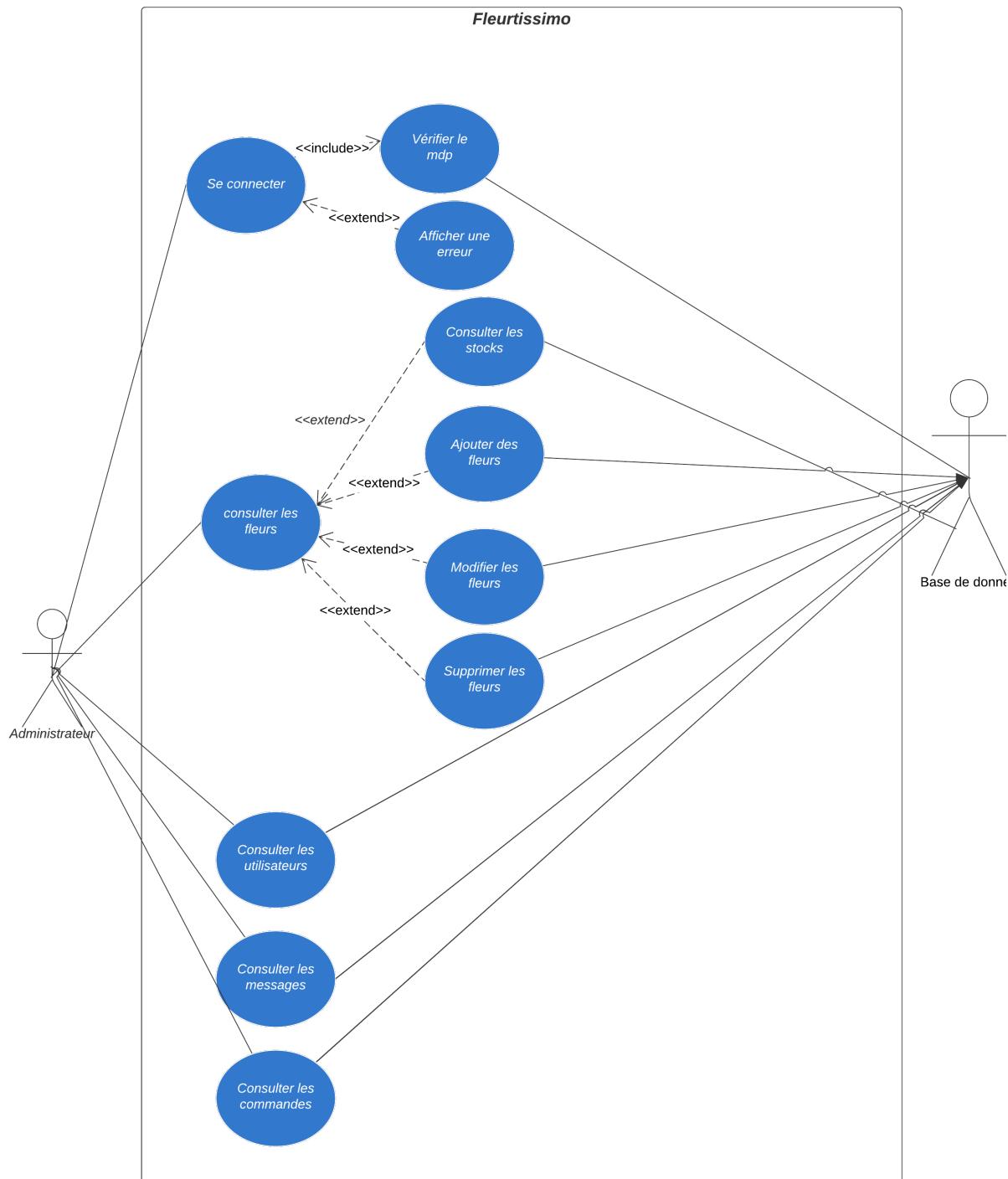


FIGURE 4: CAS D'UTILISATION MONTRANT LES INTERACTIONS DE L'ADMINISTRATEUR

2. Image de présentation de l'application

Page d'accueil

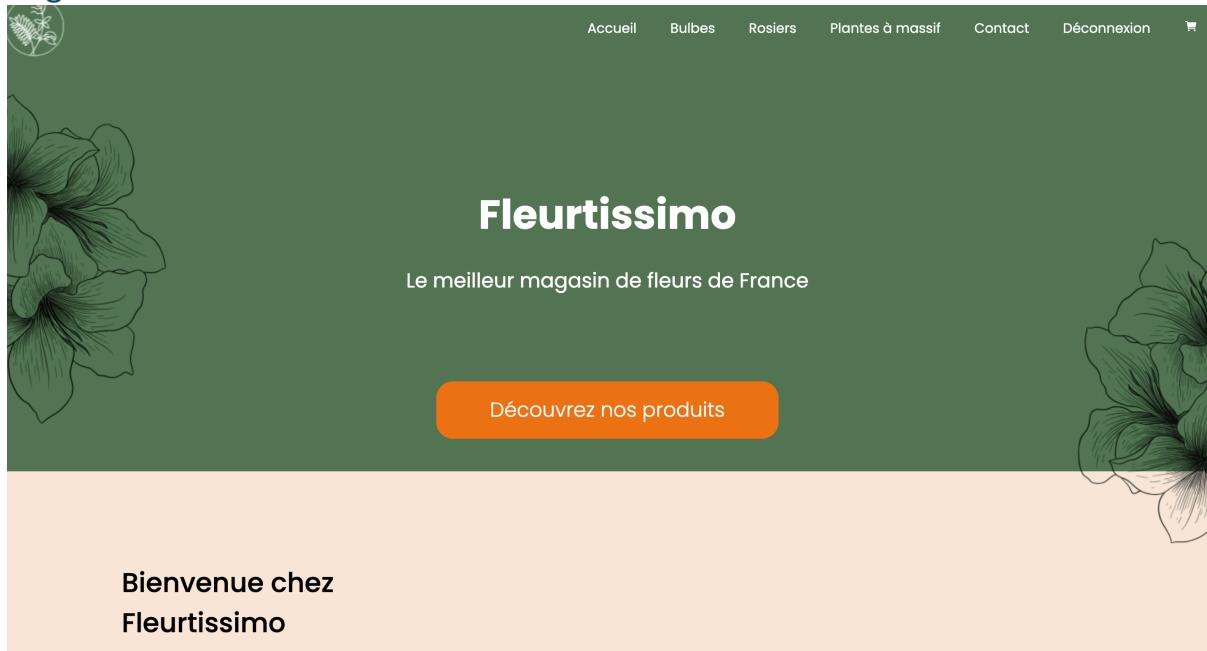


FIGURE 5: PAGE D'ACCUEIL DE L'APPLICATION



Page d'un produit

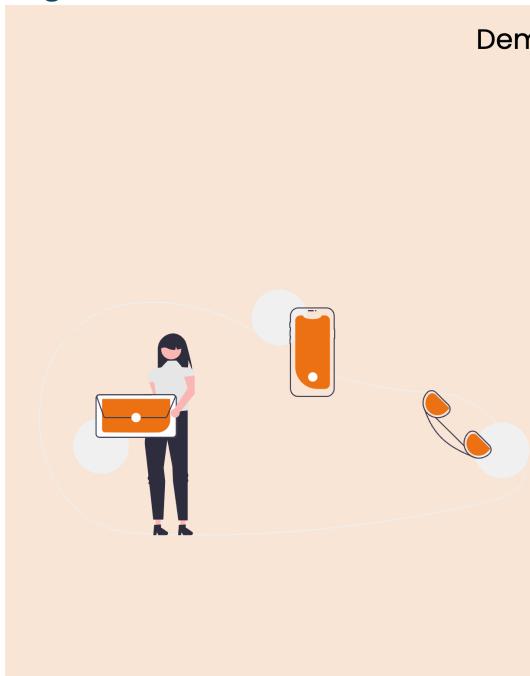
The screenshot shows a product category page with a dark green header bar containing a logo, navigation links (Accueil, Bulbes, Rosiers, Plantes à massif, Contact, Déconnexion), and a shopping cart icon. The main title 'Les bulbes' is centered above two product cards. The first card features a pink and yellow hyacinth flower cluster and is labeled 'Tulipe' with a price of '10.00 \$'. It includes a quantity selector set to '2' and a large orange 'Ajouter au panier' button. The second card features a purple allium flower cluster and is labeled 'Allium' with a price of '5.00 \$'. It includes a quantity selector set to '0' and a grey 'Ajouter au panier' button. Both cards have a small 'Stock' indicator at the bottom right.

FIGURE 6: PAGE D'UNE CATEGORIE DE PRODUIT

This is a mobile-oriented view of the same product category page. The header is a dark green bar with a circular logo on the left and a three-line menu icon in the center. Below the header, the title 'Les bulbes' is displayed. The first product card for 'Tulipe' is shown with its image, name, price '10.00 \$', quantity selector (set to 0), and a grey 'Ajouter au panier' button. A small 'Stock' indicator is visible below the button. The overall layout is designed for touch interaction on a smartphone screen.

Page de demande de contact

Demande de contact



Prénom

Nom

Email

Genre
 Homme
 Femme

Date de naissance

Fonction

Sujet

Contenu

Envoyer

FIGURE 7: PAGE DE FORMULAIRE DE CONTACT



The mobile view of the contact form page features a dark green header bar with a circular logo containing a stylized plant and a three-line menu icon. The main content area has a light orange background and displays the same form fields as the desktop version, including input fields for name, email, gender, date of birth, function, subject, and message content, along with a large orange "Envoyer" button at the bottom.

Page du panier

The screenshot shows a user's shopping cart on a website. At the top, there is a navigation bar with links for Accueil, Bulbes, Rosiers, Plantes à massif, Contact, Déconnexion, and a shopping cart icon. The main title "Panier" is centered above the cart items. Two items are listed: "Tulipe" and "Anémones du japon". Each item has a small thumbnail image, the name, the price (10.00), the quantity (2), and an orange "Supprimer" button. To the right of the items, the total price "Prix totale : 40 €" is displayed, followed by an orange "Commander" button.

FIGURE 8: PAGE DU PANIER DE L'UTILISATEUR

The mobile screenshot shows the same shopping cart page as the desktop version, but it is displayed on a smaller screen. The layout is similar, with the navigation bar at the top, the "Panier" title, and the two items in the cart. The "Tulipe" item is shown with its image, name, price (10.00), quantity (2), and "Supprimer" button. The "Anémones du japon" item is also shown with its image, name, price (10.00), quantity (2), and "Supprimer" button. The overall design is responsive and suitable for mobile devices.

Page de connexion

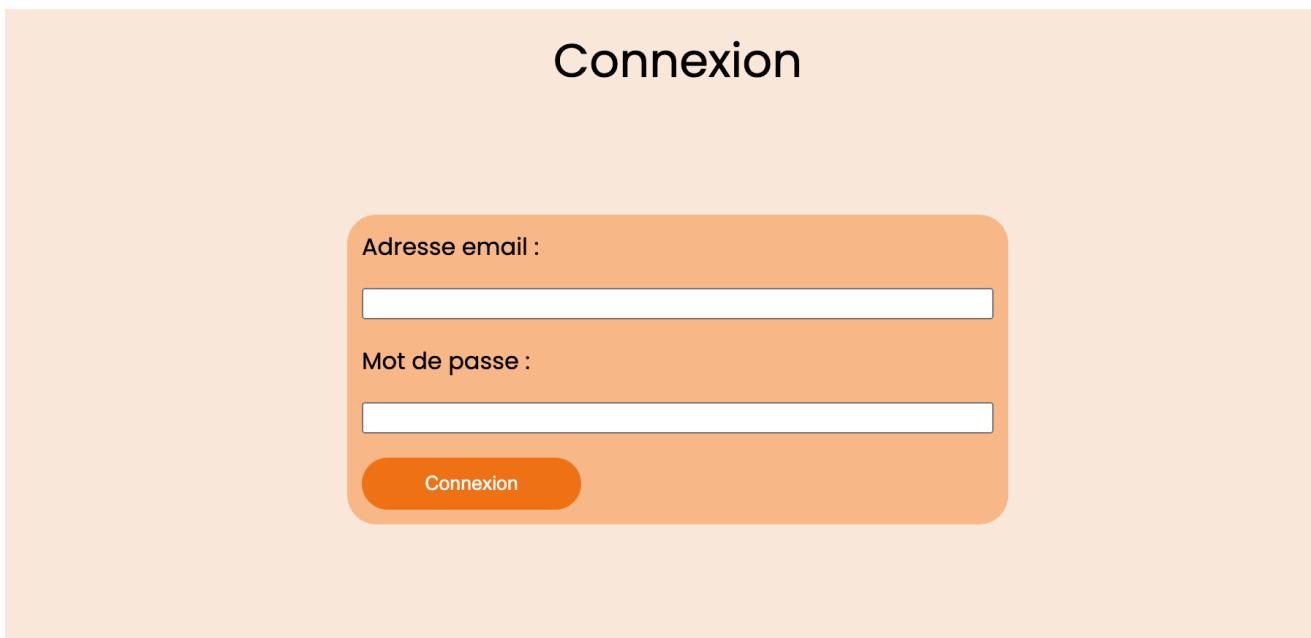
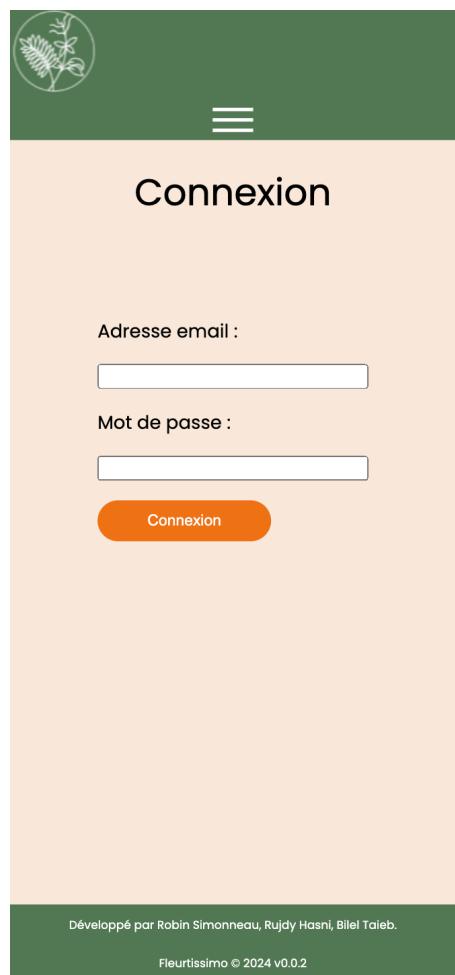


FIGURE 9: PAGE DE CONNEXION DE L'UTILISATEUR



Page d'inscription

Inscription

Nom :

Prénom :

Adresse email :

Mot de passe :

Inscription

FIGURE 10: PAGE D'INSCRIPTION DE L'UTILISATEUR

