
Internet des objets

- LEDify -

TRONTIN Pauline
MARCOURT Jean-François



Promotion 2023/2024

Table des matières

I. Phase 1 : Conception du projet	3
1. Titre	3
2. Description	3
3. Liste des fonctionnalités	3
4. Matériels/technologies	4
5. Diagrammes	4
6. Maquette.....	5
II. Phase 2 : Définition de l'API Rest (WS).....	7
7. La description des routes et les échanges de données	7
III. Phase 3 : Réalisation de l'objet (non connecté).....	9
8. La liste du matériel nécessaire	9
9. Le plan de montage de votre prototype	9
10. Le code source (lien)	10
IV. Phase 4 : Réalisation de l'objet connecté	11
11. Une explication synthétique mais documentée sur la façon dont vous avez décidé de communiquer avec l'objet (utilisation d'une librairie spécifique, de services particuliers, etc...).....	11
12. Le cas échéant, des liens vers les librairies, des services tiers que vous utilisez	11
13. Le cas échéant, une brève discussion sur les difficultés que vous avez rencontrées.	12
V. Phase 5 : Mise en place du serveur d'API et développement de l'application web	12
14. Décrire la technologie, le code métier que vous avez réalisé	12
15. Inclure une photo du produit fini, des captures d'écran de l'application.....	13
VI. Phase 6 : Démonstration du produit et rendus	15

I. Phase 1 : Conception du projet

1. Titre

Nous avons décidé de nommer notre projet « LEDify ».

2. Description

Notre projet vise à créer un affichage à LED dynamique et interactif en utilisant un ESP32. Inspiré par les panneaux d'affichage des pharmacies, notre système permettra de saisir un mot ou un message via une interface utilisateur, puis de l'afficher instantanément sur un panneau à LED.

L'ESP32 servira de cerveau du système, gérant à la fois l'interface utilisateur et le contrôle des LED. Pour cela, nous utiliserons des boutons-poussoirs ou un écran tactile comme interface utilisateur, permettant à l'utilisateur de saisir le message désiré.

Une fois le message saisi, l'ESP32 le recevra et le traitera, activant les LED appropriées pour afficher le texte. Pour simuler un effet de défilement, nous pourrions faire défiler le message de droite à gauche sur le panneau à LED, lui donnant ainsi un aspect dynamique et attrayant.

Pour assurer la flexibilité du système, nous pourrions également envisager d'ajouter des fonctionnalités supplémentaires telles que la possibilité de changer la couleur du texte, d'ajuster la vitesse de défilement ou même d'afficher des symboles ou des images simples sur le panneau à LED.

3. Liste des fonctionnalités

- Connexion à l'interface par un login
- Ajouter un bouton pour vérifier si l'ESP32 est connecté
- Affichage déroulant d'un mot
- Affichage d'une phrase/chiffre déroulante
- Contrôle luminosité des LED en fonction du potentiomètre
- Marche/Arrêt des LED en fonction du capteur de proximité
- Page Web qui envoie un mot et l'affiche en déroulant.
- Page web : on/off
- Page web : arrêt sur image (avec bouton reset)
- Change de couleur automatiquement et aléatoirement.
- Ajout d'autres fonctionnalités si on a le temps

4. Matériels/technologies

Voici la liste du matériel que l'on souhaite :

- Kit ESP 32
- 4 matrices Neopixel -> actionneur
- Un Potentiomètre -> capteur
- Capteur de proximité à ultrasons -> capteur
- Bouton
- Cable USB
- Connexion Internet

5. Diagrammes

Voici le schéma de déploiement :

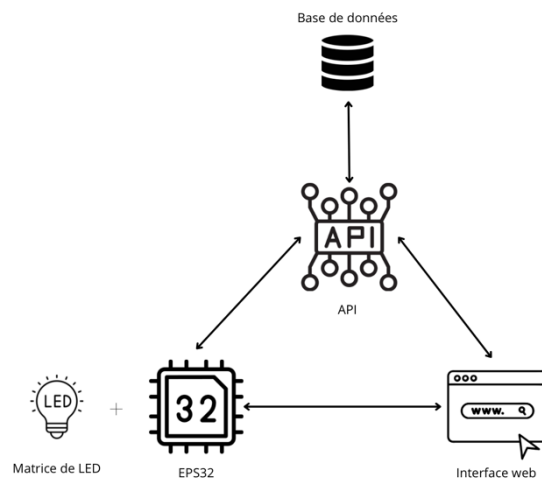
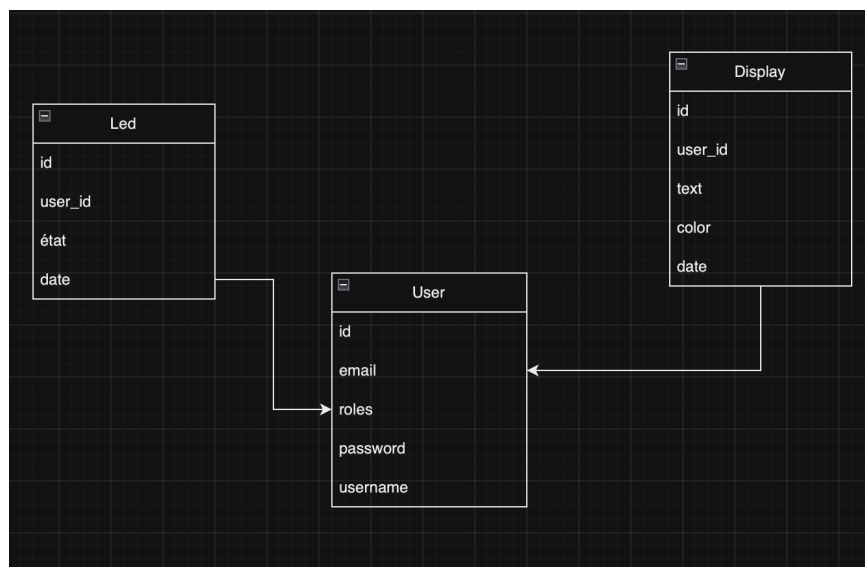
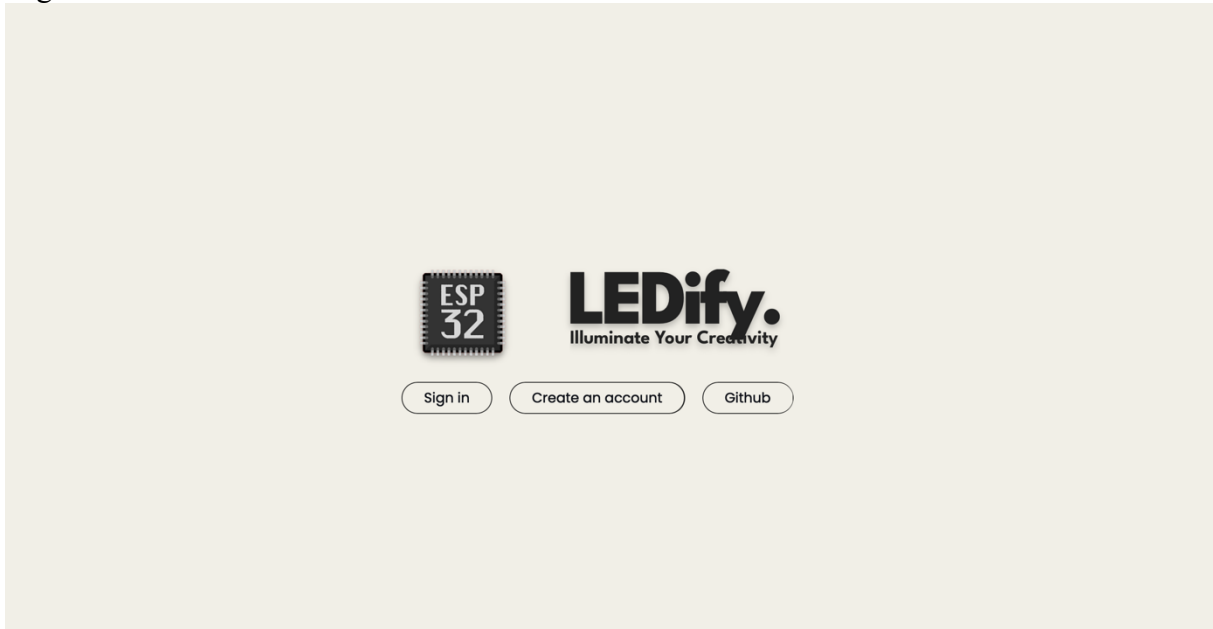


Diagramme de classe (BDD) :

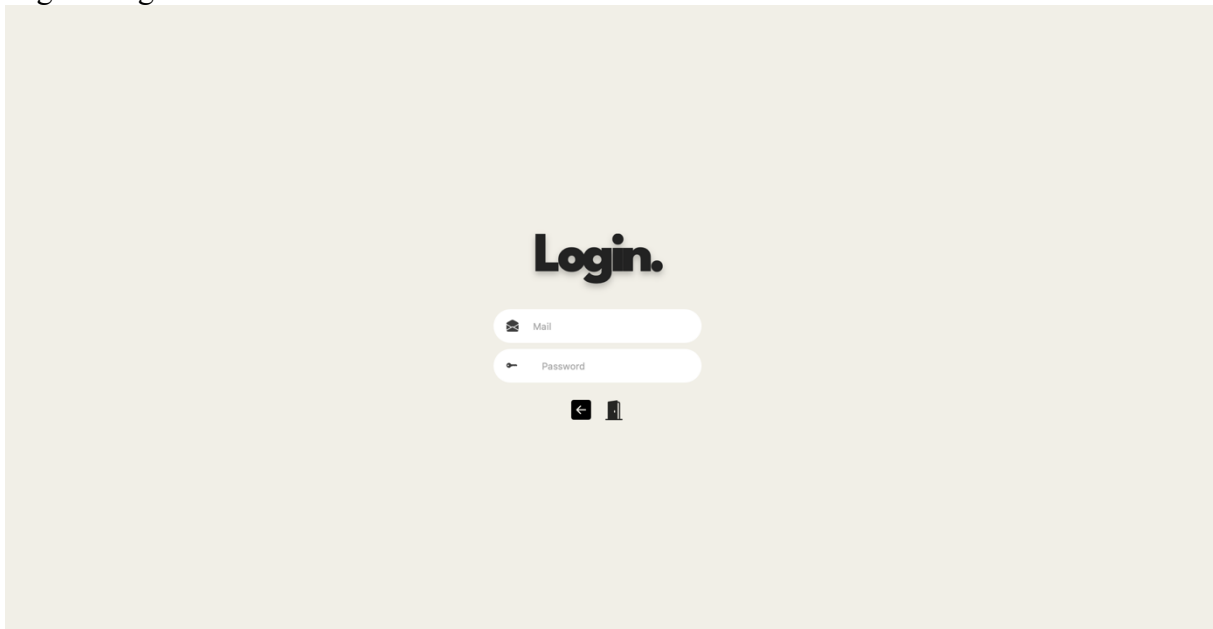


6. Maquette

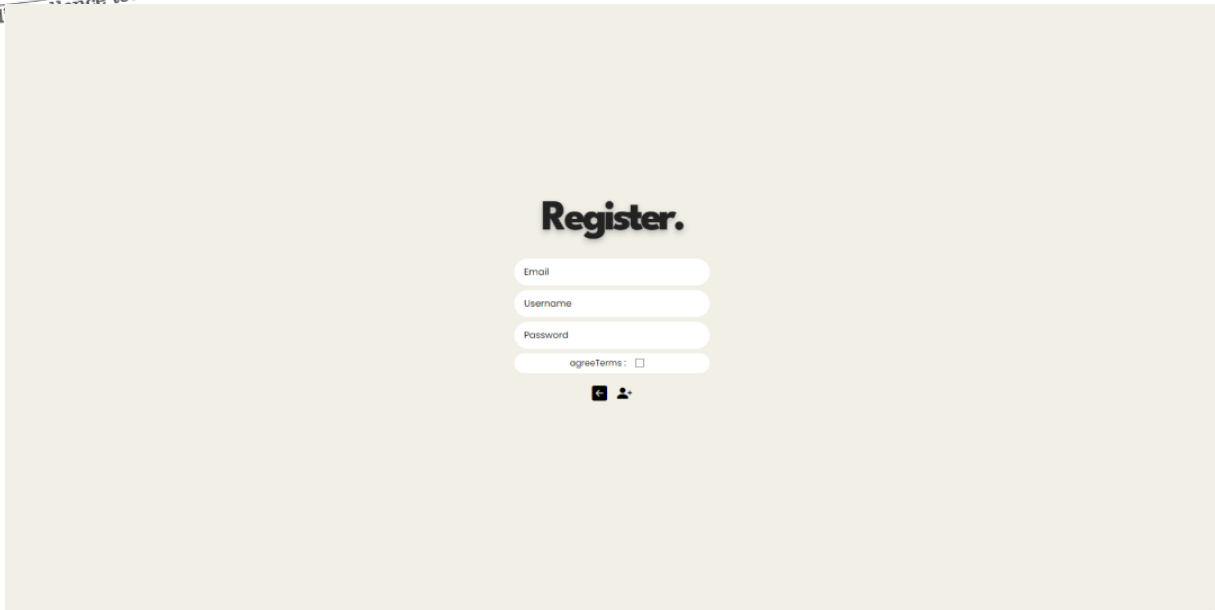
Voici les maquettes de notre futur site.
Page d'accueil :



Page de login :



Page registrer (création de compte) :





Register.

Email

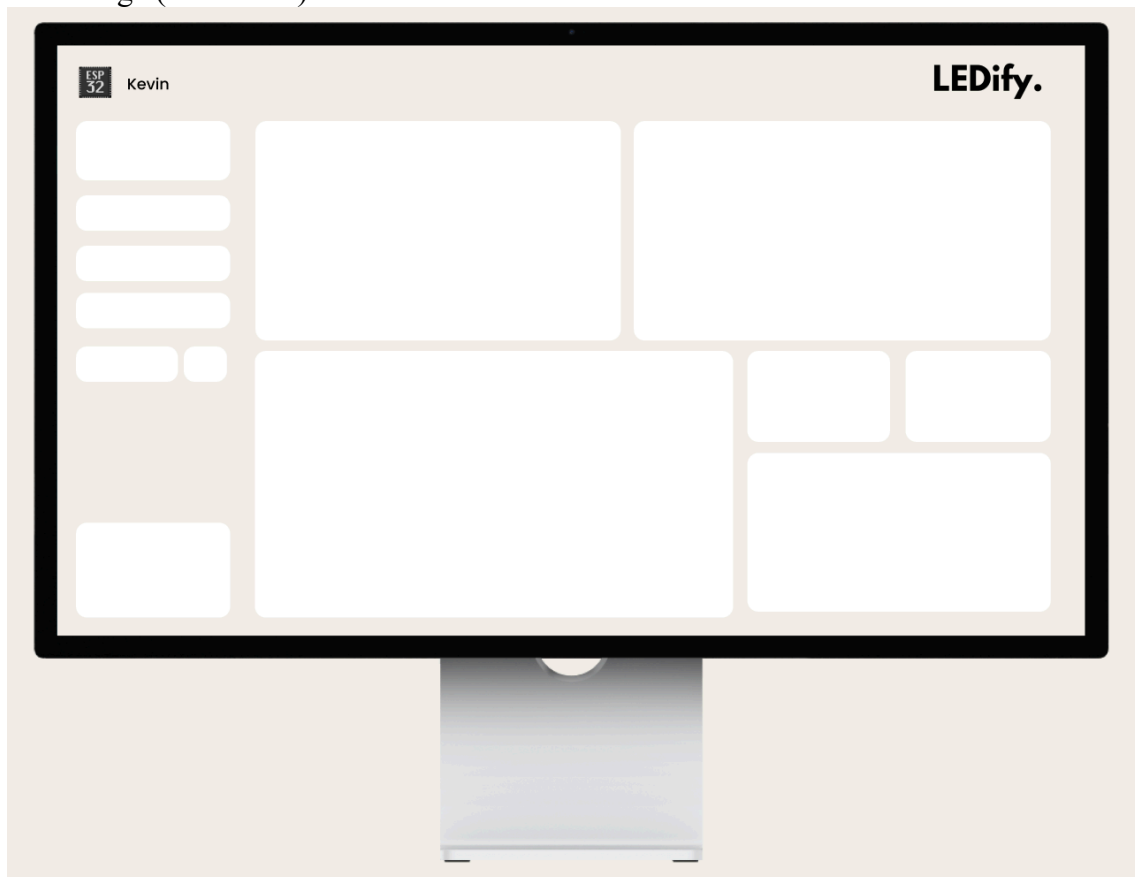
Username

Password

agreeTerms : ☐

HomePage (dashboard) :



II. Phase 2 : Définition de l'API Rest (WS)

7. La description des routes et les échanges de données

Au total nous avons une dizaine de route.

Pour accéder à la page d'accueil :

- C'est la route /
 - Elle permet d'arriver sur la page d'accueil pour se logger, créer son compte ou encore accéder au code du git.

Sur la page de login :

- /login
 - Cette route permet de se logger avec un compte déjà créée.

Sur la page registrer :

- /register
 - Cette route permet de créer un compte

Sur le home Page (page principale) :

- **ROUTE** /home
 - **URL** : / home
 - **Méthodes http** : GET
 - **Description** : Affiche la page d'accueil de l'utilisateur une fois connecté
 - **Données envoyées** : action : on ou off
 - **Fonctionnement** :
 - Vérifie si l'utilisateur est connecté.
 - Envoie une requête à l'ESP32 pour allumer ou éteindre les leds.
 - Si la commande est réussie, enregistre l'état en BDD.
 - Puis redirige vers la HomePage
- **ROUTE** /control-led
 - **URL** : /control-led
 - **Méthodes http** : GET, POST
 - **Description** : Permet de contrôler l'état des LEDs en envoyant une commande pour les allumer ou les éteindre.
 - **Données envoyées** : action : on ou off
 - **Fonctionnement** :
 - Vérifie si l'utilisateur est connecté.
 - Envoie une requête à l'ESP32 pour allumer ou éteindre les leds.
 - Si la commande est réussie, enregistre l'état en BDD.
 - Puis redirige vers la HomePage
- **ROUTE** /display_word_and_color

- **URL :** /display_word_and_color
- **Méthodes http :** GET, POST
- **Description :** Permet d'afficher et de faire défiler un mot avec une couleur choisi sur la matrice de LED.
- **Données envoyées :**
 - Word : le mot à afficher
 - Color : la couleur du mot à afficher
- **Fonctionnement :**
 - Vérifie si l'utilisateur est connecté.
 - Envoie le mot et la couleur à l'ESP32
 - Si l'envoi est réussie, enregistre les détails et l'horodatage en BDD.
 - Puis redirige vers la HomePage avec un message de succès.
- **ROUTE /color**
 - **URL :** /color
 - **Méthodes http :** GET, POST
 - **Description :** Permet de changer la couleur total des LEDs.
 - **Données envoyées :**
 - Color : la couleur à afficher (en format hexadécimal)
 - **Fonctionnement :**
 - Envoie la nouvelle couleur à l'ESP32
 - Affiche un message d'erreur ou de succès en fonction de la réponse de l'ESP32.
 - Puis redirige vers la HomePage
- **ROUTE /mode**
 - **URL :** /mode
 - **Méthodes http :** GET, POST
 - **Description :** Permet de choisir un mode d'affichage pour les LEDs.
 - **Données envoyées :**
 - Mode : le mode sélectionnée (ex : 1,2,3....)
 - **Fonctionnement :**
 - Envoie le mode sélectionnée à l'ESP32
 - Affiche un message d'erreur ou de succès en fonction de la réponse de l'ESP32.
 - Puis redirige vers la HomePage
- **ROUTE /interrupteur**
 - **URL :** / interrupteur
 - **Méthodes http :** GET
 - **Description :** Permet de d'afficher l'état actyel de la LEd pour l'utilisateur connecté.
 - **Fonctionnement :**

- Récupère l'état le plus récent de la LED en base de données pour l'utilisateur connecté
- Affiche l'état de la LED dans le Template.
- **ROUTE** /interrupteur
 - **URL** : / interrupteur
 - **Méthodes http** : GET
 - **Description** : Permet de d'afficher l'état actuel de la LED pour l'utilisateur connecté.
 - **Fonctionnement** :
 - Récupère l'état le plus récent de la LED en base de données pour l'utilisateur connecté
 - Affiche l'état de la LED dans le Template.

III. Phase 3 : Réalisation de l'objet (non connecté)

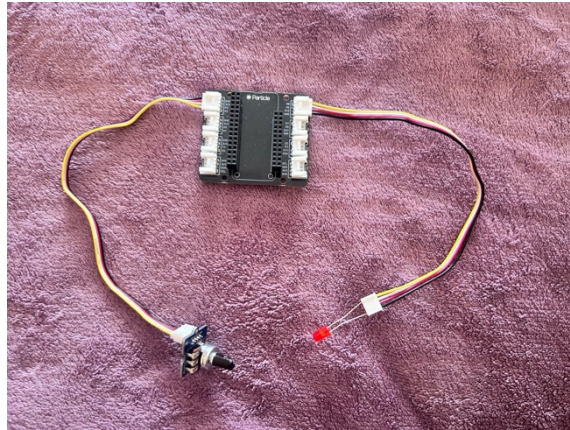
8. La liste du matériel nécessaire

Voici la liste du matériel après ajustement :

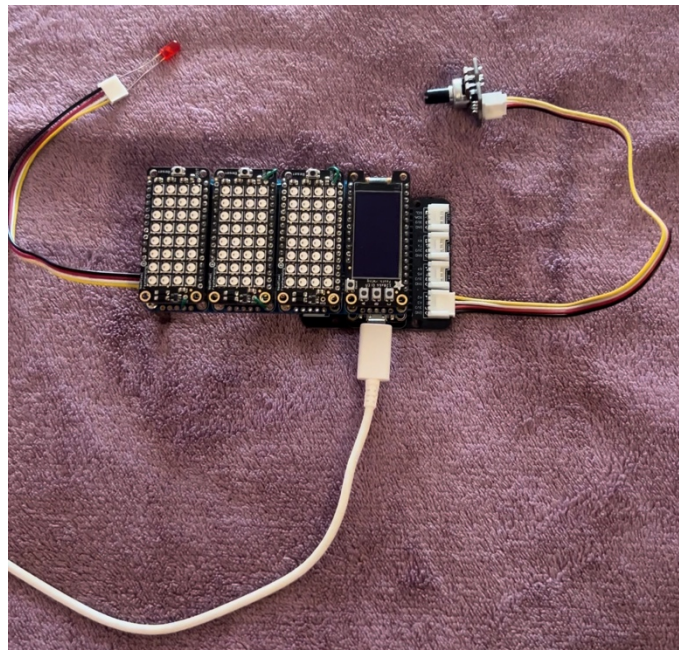
- Kit ESP 32
- 4 matrices Neopixel -> actionneur
- Un Potentiomètre -> capteur
- Capteur de proximité à ultrasons -> capteur
- Bouton
- Petite lumière
- Cable USB
- Carte Wifi
- Connexion Internet

9. Le plan de montage de votre prototype

Tout d'abord il faut brancher le potentiomètre sur le PIN A2 et la petite led externe sur le PIN A0 (voir capture 1). Puis brancher l'ESP32 sur le plug. Puis venez ajouter la matrice de LED par-dessus. Ensuite vous avez plus qu'à relier la carte à votre ordinateur grâce à un câble USB (voir capture 2).



Voir capture 1



Voir capture 2

10. Le code source (lien)

Notre code source est sur le gitlab de l'IUT, voici le lien :

<https://iutbg-gitlab.iutbourg.univ-lyon1.fr/but3-wot-23-24/trontin-pauline-marcourt-jf>

IV. Phase 4 : Réalisation de l'objet connecté

11. Une explication synthétique mais documentée sur la façon dont vous avez décidé de communiquer avec l'objet (utilisation d'une librairie spécifique, de services particuliers, etc...)

Pour communiquer avec l'objet, nous avons opté pour une approche basée sur le protocole HTTP via un serveur web embarqué sur l'ESP32, ce qui permet par exemple de contrôler l'allumage et l'extinction d'une matrice de LED à distance. Pour cela, nous avons utilisé les bibliothèques AsyncTCP et ESPAsyncWebServer pour gérer les requêtes HTTP de manière asynchrone.

Le code Arduino intègre également la librairie WiFi pour faciliter la connexion à un réseau WiFi, assurant ainsi une mise en place rapide et efficace. De plus, nous avons inclus les bibliothèques Adafruit_GFX, Adafruit_NeoPixel et Adafruit_NeoMatrix pour la manipulation et l'affichage des LED sur la matrice.

La communication avec l'ESP32 se fait via des requêtes HTTP GET et POST envoyées depuis un site Symfony. Par exemple, l'action d'allumer ou d'éteindre les LED est déclenchée en fonction des paramètres transmis dans l'URL. Cette approche offre une interface conviviale et flexible pour interagir avec l'objet depuis n'importe quel appareil connecté au même réseau WiFi.

12. Le cas échéant, des liens vers les librairies, des services tiers que vous utilisez

Voici les librairies utilisées :

- « Wifi »
 - <https://www.arduino.cc/reference/en/libraries/wifi/>
- « AsyncTCP »
 - <https://www.arduino.cc/reference/en/libraries/asynctcp/>
- « ESPAsyncWebServer »
 - <https://www.arduino.cc/reference/en/libraries/espasyncwebserver/>
- « Adafruit_GFX »
 - <https://www.arduino.cc/reference/en/libraries/adafruit-gfx-library/>
- « Adafruit_NeoMatrix »
 - <https://www.arduino.cc/reference/en/libraries/adafruit-neomatrix/>
- « Adafruit_NeoPixel »
 - <https://www.arduino.cc/reference/en/libraries/adafruit-neopixel/>

13. Le cas échéant, une brève discussion sur les difficultés que vous avez rencontrées.

La principale difficulté rencontrée lors du développement du projet a été la mise en place de la connexion au réseau WiFi de l'ESP32. En effet, une contrainte inattendue est survenue lors des tests avec des appareils iOS, notamment les iPhones. Il s'avère que certains paramètres de sécurité sur les appareils iOS peuvent restreindre ou bloquer la connexion automatique à des réseaux WiFi configurés dynamiquement. Ce qui nous a fait perdre énormément de temps car en cours nous ne pouvions pas travailler.

V. Phase 5 : Mise en place du serveur d'API et développement de l'application web

14. Décrire la technologie, le code métier que vous avez réalisé

Dans le cadre de la réalisation du projet il nous a été question de choisir une technologie de développement afin de pouvoir construire le plus proprement et efficacement possible l'API, ainsi que l'interface web qui allait par la suite être utilisée pour interagir avec celle-ci.

Étant de fidèles partisans, et développeurs habitués de technologie PHP notamment sous le Framework Symfony, utilisé depuis quelques temps par les deux membres de notre binôme, en entreprise, à l'université, ainsi que pour des projets personnels, la décision concernant le choix de la technologie fut rapide à prendre.

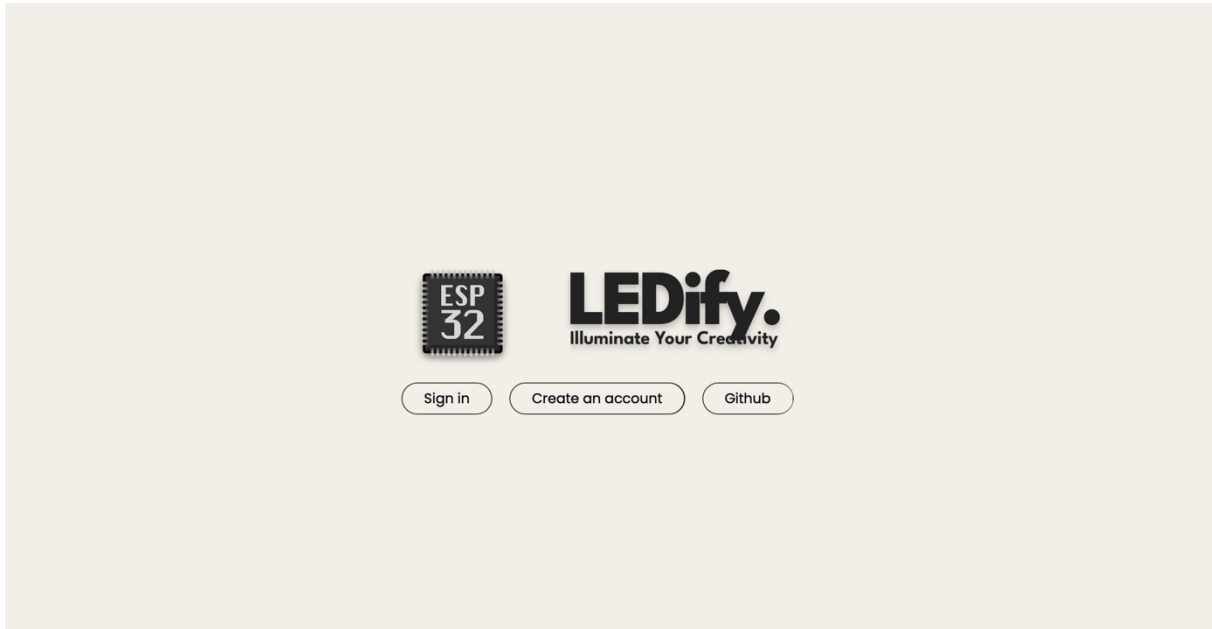
Symfony est particulièrement adapté pour ce type de projet. Le Framework fonctionne de sorte à nécessiter nativement une API pour fonctionner ainsi nous n'avons pas besoin de développer simultanément deux applications web, une seule suffit, comprenant l'API backend, ainsi que les différentes pages de l'interface web qui nous permet d'appeler facilement les différentes routes de celle-ci.

Pour accélérer le développement de notre interface, et pouvoir se concentrer sur les priorités de développement, nous avons réutilisé la base Symfony que nous avons développé ensemble sur un précédent projet.

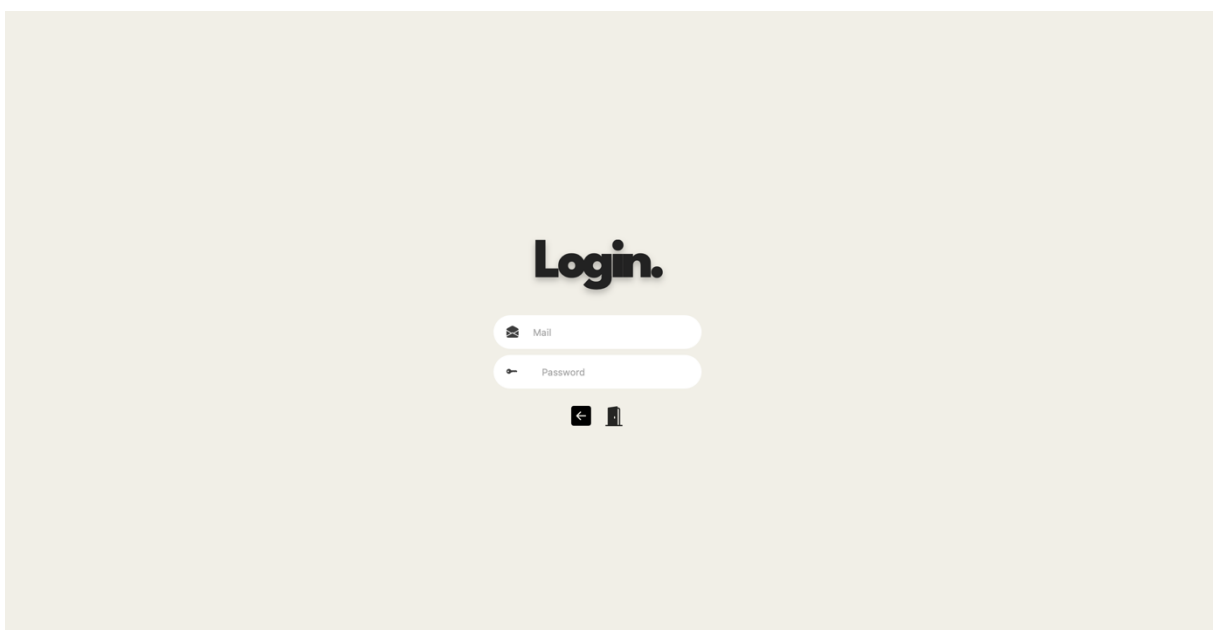
Ainsi nous avons pu rapidement nous mettre au travail, et développer l'API en parallèle du code du micro-Controller, en partant sur une base de code que nous connaissions sur le bout des doigts, d'un langage idéal pour le projet.

15. Inclure une photo du produit fini, des captures d'écran de l'application

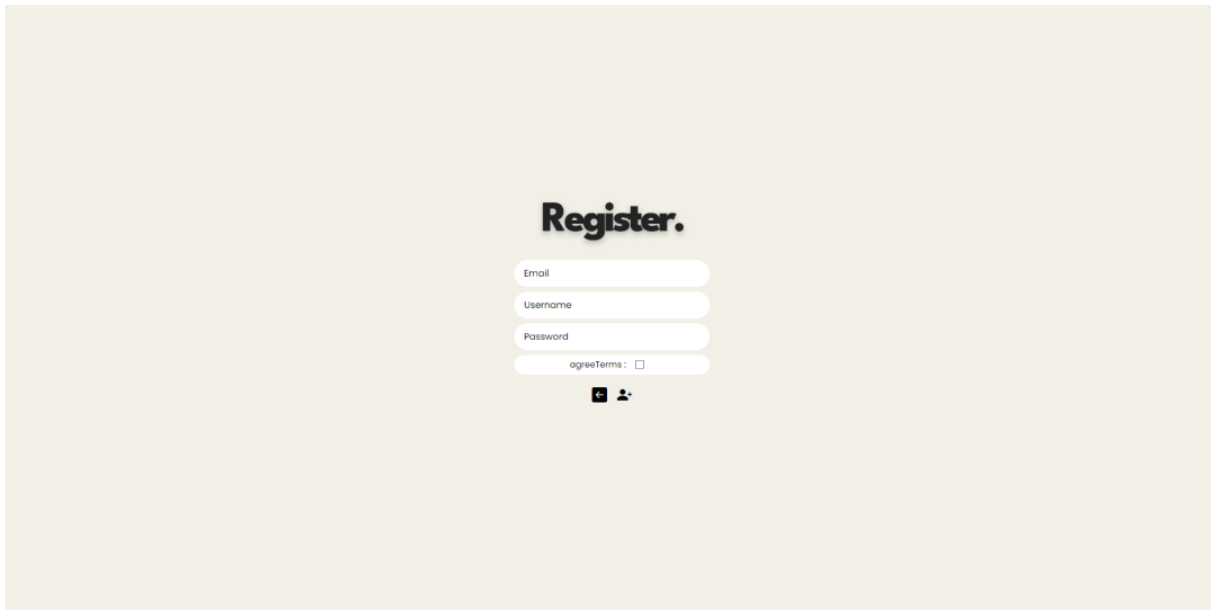
Voici la page d'accueil (route /accueil) :



Voici la page du login (route /login) :



Voici la page du register (route /register) :





Register.

Email

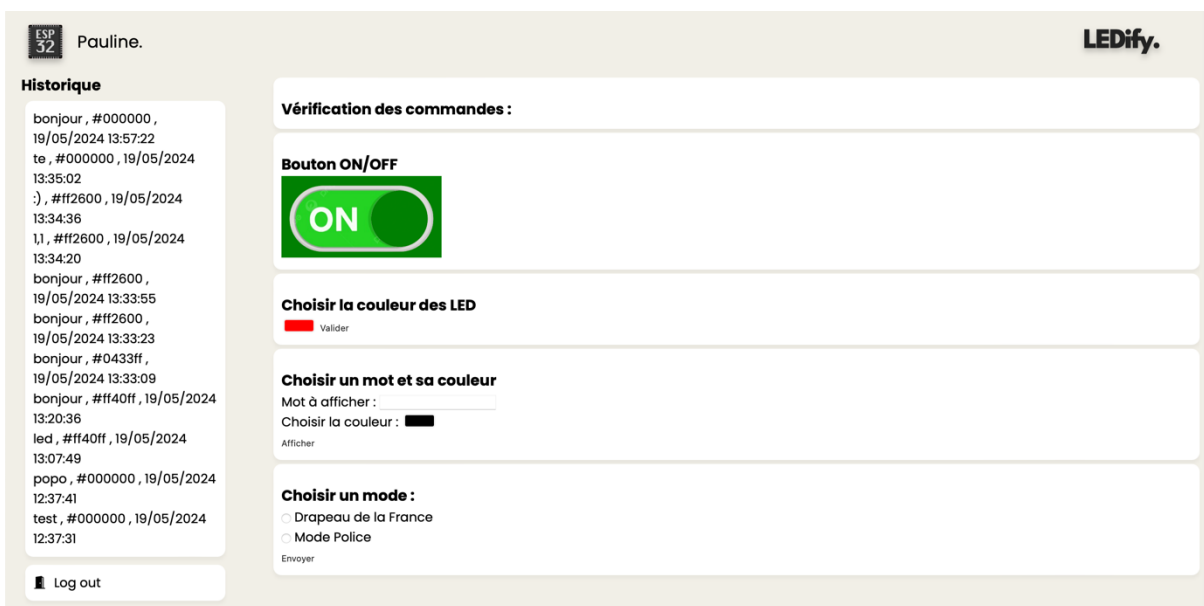
Username

Password

agreeTerms : ☐


Voici la HomePage (route /home) :



ESP 32 Pauline. **LEDify.**


Historique

bonjour , #000000 ,
19/05/2024 13:57:22
te , #000000 , 19/05/2024
13:35:02
:) , #ff2600 , 19/05/2024
13:34:36
l,l , #ff2600 , 19/05/2024
13:34:20
bonjour , #ff2600 ,
19/05/2024 13:33:55
bonjour , #ff2600 ,
19/05/2024 13:33:23
bonjour , #0433ff ,
19/05/2024 13:33:09
bonjour , #ff40ff , 19/05/2024
13:20:36
led , #ff40ff , 19/05/2024
13:07:49
popo , #000000 , 19/05/2024
12:37:41
test , #000000 , 19/05/2024
12:37:31


 Log out

Vérification des commandes :

Bouton ON/OFF




Choisir la couleur des LED

 Valider

Choisir un mot et sa couleur

Mot à afficher :

Choisir la couleur : 

Afficher

Choisir un mode :

☐ Drapeau de la France

☐ Mode Police

Envoyer

VI. Phase 6 : Démonstration du produit et rendus

Lien de la vidéo CANVA :

[https://www.canva.com/design/DAGFrSLBQ70/k7re-](https://www.canva.com/design/DAGFrSLBQ70/k7re-FIfQiUBJVPUeU1Sxg/watch?utm_content=DAGFrSLBQ70&utm_campaign=designshare&utm_medium=link&utm_source=editor)

[FIfQiUBJVPUeU1Sxg/watch?utm_content=DAGFrSLBQ70&utm_campaign=designshare&utm_medium=link&utm_source=editor](https://www.canva.com/design/DAGFrSLBQ70/k7re-FIfQiUBJVPUeU1Sxg/watch?utm_content=DAGFrSLBQ70&utm_campaign=designshare&utm_medium=link&utm_source=editor)