

Meca Geoffrey

Concepteur développeur d'application

Dossier Projet - Code hub

2023

Sommaire

Concevoir et développer des composants d'interface utilisateur

Présentation du projet	4
Le projet	4
Travail en équipes	5
Apprentissages techniques	6
La maquette du projet	6
Conception de la maquette	6
Réunion préparatoire	6
Le choix d'un nuancier de couleur	7
Maquette	8
La landing page	9
L'inscription	9
Liste des articles	10
Article	10
Ajouter un article	11
Choix de la navigation	11

Concevoir et développer la persistance des données

	12
Présentation base de données	12
Le MCD	13
Le MLD	13
Description de l'API REST	14
Les entités	14
Entité user	15
State Provider	15
State Processor	16
Tests Unitaire	17
Tests Fonctionnel	19

Concevoir et développer une application multicouche

Les fonctionnalités du projet Code hub	21
Utilisateur non connecté	21
Création d'un compte utilisateur	21

Les articles	22
Les commentaires	23
Utilisateur connecté	24
Administration de son compte utilisateur	25
Création d'articles	26
Création d'un commentaire	27
Administrateur du site	28
Modération des articles	29
Suppression des articles	29
Modération des commentaires	29
Suppression des commentaires	29
Back-end	30
Ajout et appelle d'une route	26
Projet d'avenir et prochaines fonctionnalités	31
Conclusion	32

Présentation du projet

Le projet Code hub se base sur plusieurs objectifs. Il a été développé par une équipe de quatre développeurs, dans le cadre de notre formation en tant que “Concepteur développeur d’application”. Ce projet est le point d’orgue de notre formation et nous a apporté diverses expériences dans la mise en place, la conception et la création d’une application.

Le Projet

Le forum Code Hub est une application mobile Android, développée pour permettre aux utilisateurs de publier des articles et de commenter ceux des autres.

L'application permet aux visiteurs sans compte de consulter les articles et les commentaires, mais ils ne peuvent pas en ajouter eux-mêmes.

Les utilisateurs avec un compte peuvent ajouter des articles et des commentaires. Ils ont également la possibilité de modifier leur profil et de supprimer leur compte si nécessaire.

Les administrateurs ont les mêmes droits qu'un utilisateur standard, mais disposent également de fonctionnalités supplémentaires.

Ils peuvent modifier, supprimer et ajouter des utilisateurs au forum. De plus, ils peuvent supprimer des articles ou des commentaires en cas de violation des règles du forum.

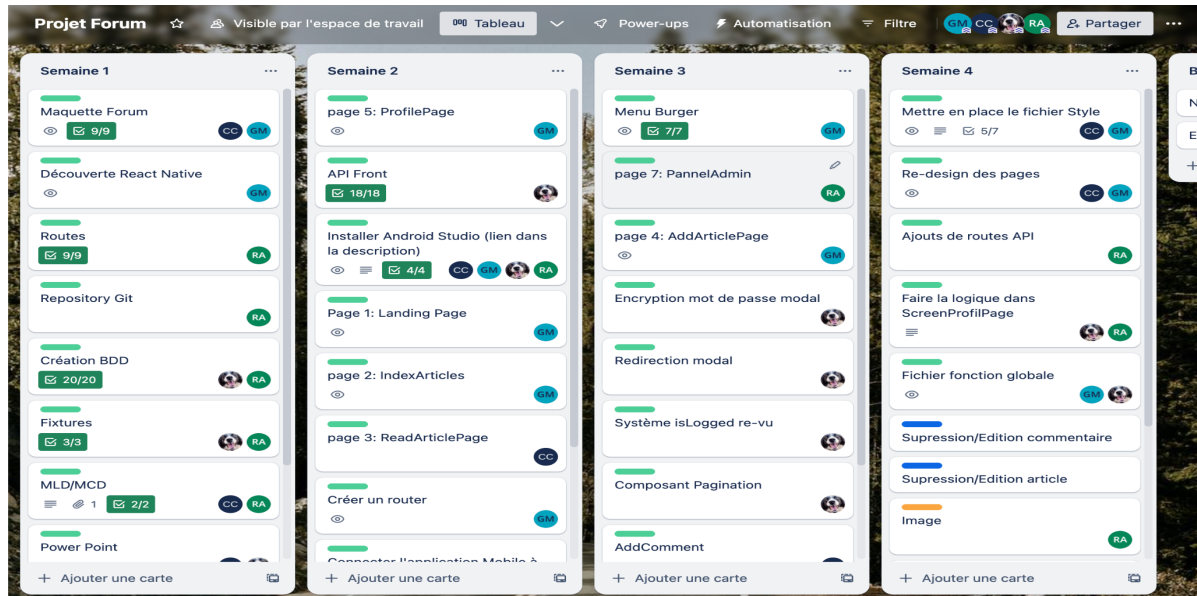
Il possède plusieurs fonctionnalités :

- Lecture d'article,
- Lecture des commentaires,
- Modération ou suppression des articles,
- Modération ou suppression des commentaires,
- Création d'un compte utilisateur,
- Administration de son compte utilisateur,
- Création d'articles si l'on est connecté en tant qu'utilisateur,
- Création de commentaires sous les articles si l'on est connecté en tant qu'utilisateur,
- Administration de son profil utilisateur,
- Possibilité d'effacer son compte utilisateur

Travail en équipes

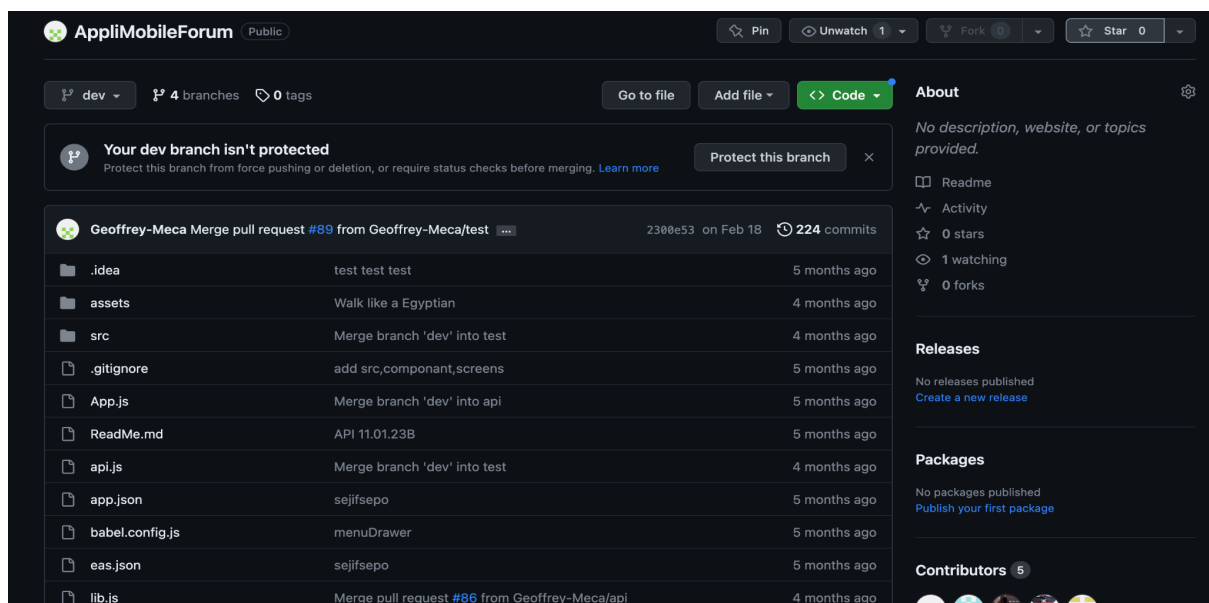
La première étape dans la réalisation d'un projet informatique est de construire une équipe solide de développeurs qui peuvent collaborer efficacement pour atteindre les objectifs du projet.

Nous avons constitué une équipe de quatre développeurs et utilisé les méthodes de **Kanban** et **Trello** pour le suivi des tâches.



Le projet a été divisé en petites tâches pour faciliter le travail et assurer une bonne gestion du temps.

Nous avons également utilisé **Github** pour le suivi des versions et le partage du code.



Apprentissages techniques

Le projet Code hub s'appuie sur plusieurs types de technologies, notamment, les API REST créées à l'aide du Frameworks Symfony et API plateforme, la création d'applications mobiles avec React Native et la librairie Axios.

Ces éléments ont été pour toute l'équipe, une opportunité de monter en compétence sur ces technologies.

La maquette

Conception de la maquette

Code hub est un projet commun, il a donc était nécessaire dans un premier temps de créer une maquette de l'application qui réponde à la fois au cahier des charges qui nous a été soumis par La plateforme, mais aussi qu'elle réponde à des critères esthétiques, veillé à ce que la lisibilité soit bonne, vu que nous avons créer un forum, une application permettant de lire des articles, les écritures ou les commenter.

Nous nous sommes donc appuyés sur divers critères pour créer la maquette qui nous a guidé durant tout le développement de notre application.

Réunion préparatoire

La réunion préparatoire a permis d'établir la structure du site, nous étions quatre à apporter des idées et à en débattre. Chacun avec sa sensibilité et son vécu en tant que développeur, nous avons donc établi diverses possibilités. Nous avons définis ensemble un parcours utilisateur ainsi qu'établi pour chaque fonctionnalité de l'application, la forme qu'elle devait prendre.

Le choix d'un nuancier de couleur

Le choix de couleur a été essentiellement définis pour rendre la lecture des articles lisible, mais proposé une touche d'originalité, le choix du bleu majoritaire et du blanc pour l'écriture a vite été adopté par l'équipe.



Wireframe

Le Wireframe a permis de mettre au propre les idées de chacun et de créer un parcours utilisateur simplifié sans s'attacher à une réalisation de maquette très détaillée. Cela a permis aussi de valider le parcours utilisateur de manière intuitive et simplifiée.



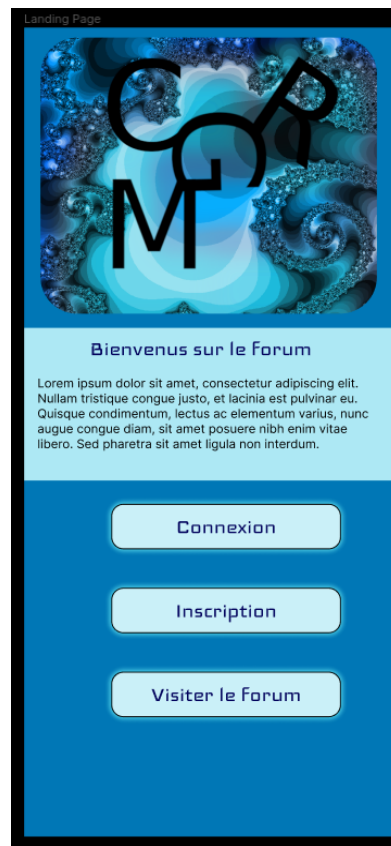
Maquette

La maquette a été réalisée une fois que le wireframe a été validé par l'ensemble des membres de l'équipe.

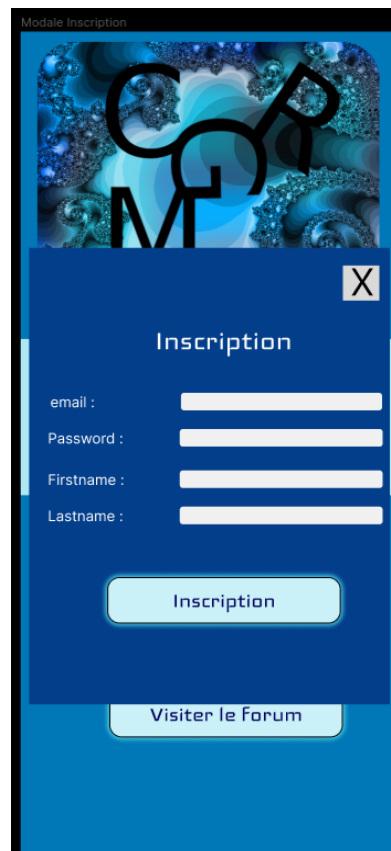
La maquette a été réalisée sur Figma.



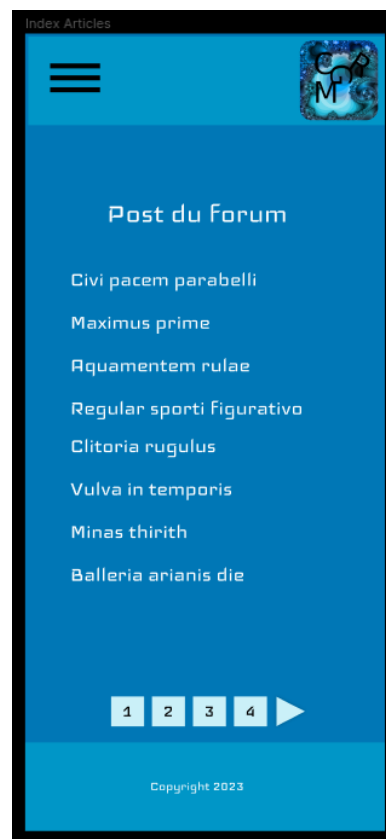
La landing page



L'inscription



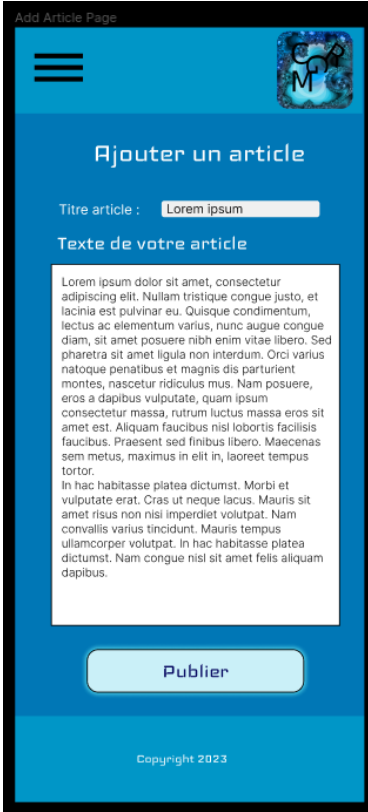
Liste des articles



Article



Ajouter un article



The screenshot shows a mobile application interface for adding a new article. At the top, there is a header bar with a hamburger menu icon on the left and a profile picture placeholder on the right. Below the header, the title 'Ajouter un article' is centered. Underneath, there is a text input field labeled 'Titre article :'. Below that, there is a section titled 'Texte de votre article' containing a large text area with placeholder Lorem Ipsum text. At the bottom of the form, there is a blue button labeled 'Publier'. The footer of the app shows 'Copyright 2023'.

Choix de la navigation

La navigation est un aspect important dans le développement des applications mobiles car elle permet aux utilisateurs de naviguer facilement entre les différentes pages et fonctionnalités de l'application. React Native offre plusieurs options pour gérer la navigation, chacune ayant ses avantages et inconvénients.

La première option de navigation en React Native est la navigation basée sur des piles (stack navigation). Elle permet d'empiler des écrans les uns sur les autres et de naviguer entre eux en utilisant des boutons de navigation ou des gestes de balayage. Cette méthode est idéale pour les applications qui ont une structure de navigation simple avec une hiérarchie linéaire d'écrans.

La deuxième option de navigation est la navigation basée sur des onglets (tab navigation). Cette méthode permet d'afficher différentes sections de l'application sur des onglets, chacun ayant son propre contenu. Les utilisateurs peuvent naviguer entre les onglets en appuyant sur l'onglet souhaité. Cette méthode est idéale pour les

applications qui ont plusieurs sections distinctes qui peuvent être explorées de manière indépendante.

La troisième option de navigation est la navigation basée sur un menu latéral (drawer navigation). Cette méthode permet d'afficher un menu latéral sur le côté de l'écran, qui peut être ouvert ou fermé pour afficher différentes sections de l'application. Cette méthode est idéale pour les applications qui ont plusieurs sections différentes qui ne sont pas facilement accessibles depuis l'écran principal.

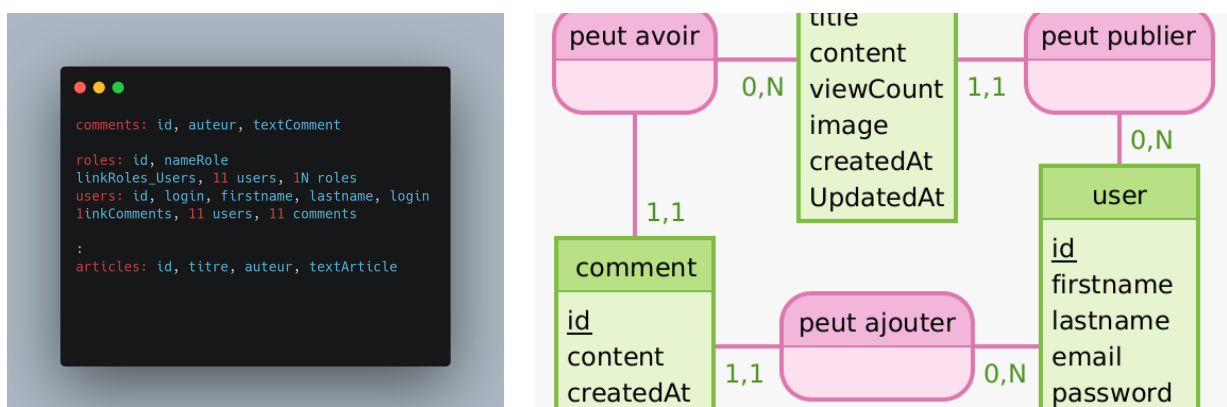
Le Drawer Navigation est une fonctionnalité de navigation couramment utilisée dans les applications mobiles pour améliorer l'expérience utilisateur, offrir un design moderne et élégant et faciliter la navigation entre les différentes sections de l'application. Pour l'utiliser, nous avons besoin d'une bibliothèque de navigation qui supporte cette fonctionnalité et nous devons implémenter le menu déroulant et la gestion de l'interaction avec l'utilisateur en utilisant les composants fournis par la bibliothèque.

Présentation base de données

La base de données a été le premier élément que nous avons dû conceptualiser. Pour ce faire, nous avons utilisé divers outils pour la mettre en œuvre. Les fonctionnalités de l'application nous ont permis de mieux comprendre la structure des tables et leur interaction dans la future application. Nous avons pris en compte l'ensemble des fonctionnalités pour concevoir les tables de la base de données ainsi que leur interaction. Nous avons ensuite créé le MCD et le MLD de la base de données pour nous assurer que tout ce que nous avons prévu était possible dans notre modèle de données. Ce travail en amont nous a permis de comprendre et anticiper le déroulement du développement de l'application future.

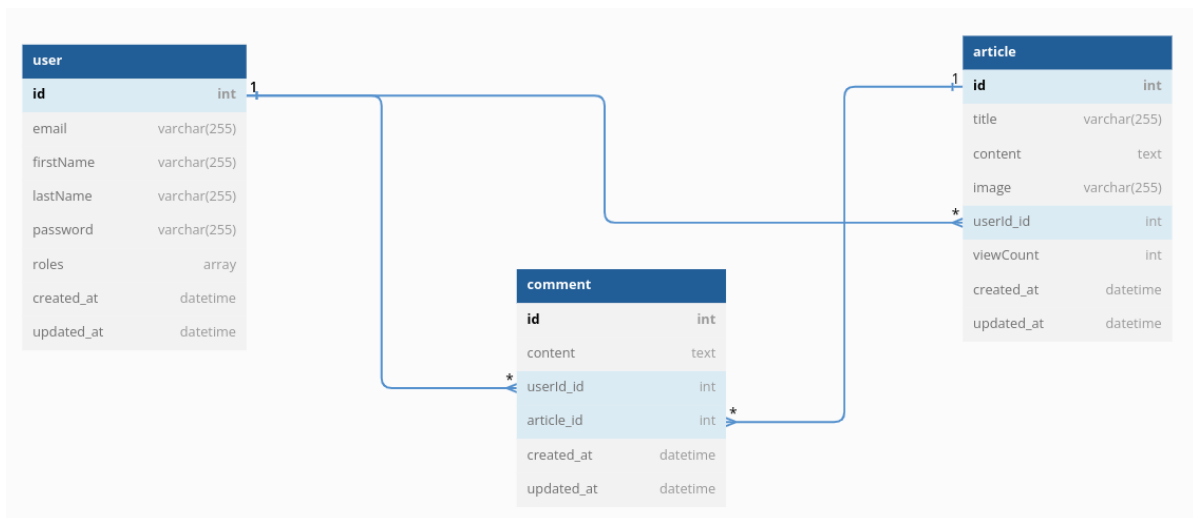
Le MCD

Le MCD (Modèle Conceptuel de données) a été créé avec l'outil en ligne Mocodo qui permet de manière simple, de créer un MCD via quelques lignes de commande et obtenir une représentation graphique claire de ce MCD. Cet outil permet de gagner du temps dans la conception d'une base de données.



Le MLD

Le Modèle Logique de données a été créé en se basant sur le MCD.

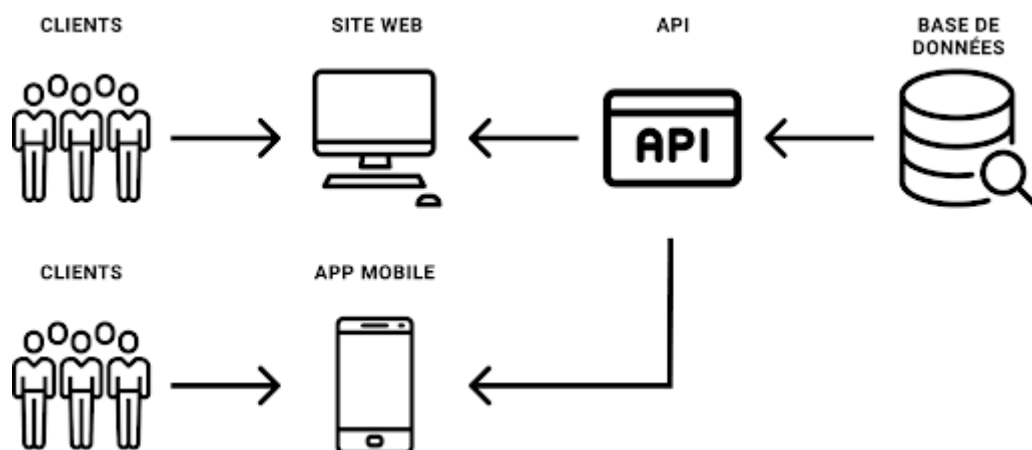


Description de l'API REST

Dans cette section, nous allons présenter notre API réalisée avec les frameworks API Platform et Symfony. Dans ce projet, nous avons utilisé ces technologies pour créer une API REST performante et facile à utiliser.

Application Programming Interface (API) : Une Application Programming Interface (API) est une façade par laquelle un logiciel fournit des services à une autre application, la plupart du temps via une interface web.

- Permet d'exposer des ressources vers l'extérieur.
- Un seul back, plusieurs front (clients)



Une API REST (Representational State Transfer) est un style d'architecture qui permet à différents systèmes de communiquer entre eux via des requêtes HTTP. Cette architecture est devenue très populaire ces dernières années grâce à sa flexibilité et à sa capacité à fonctionner avec un large éventail de langages de programmation et de systèmes.

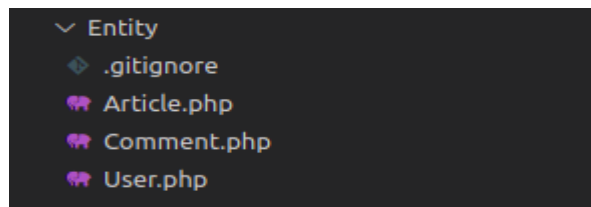
Les API REST fonctionnent en envoyant des requêtes HTTP à des ressources spécifiques, qui peuvent être des données, des fichiers ou d'autres types de ressources. Les réponses renvoyées par l'API sont généralement des données structurées dans des formats tels que JSON ou XML. Il repose sur un certain nombre de principes clés, tels que l'utilisation de verbes HTTP standard tels que GET, POST, PUT, PATCH et DELETE pour effectuer des opérations sur les ressources, l'utilisation de l'URI pour identifier de manière unique les ressources, et l'utilisation de l'état représentationnel pour représenter l'état des ressources.

Dans ce projet, nous avons utilisé API Platform, un framework web utilisé pour générer des API REST et GraphQL4, se basant sur le patron de conception MVC, tout en offrant des fonctionnalités avancées telles que la documentation automatique de l'API, la validation des données, la sécurité, etc. La partie serveur du framework est écrite en PHP et basée sur le framework Symfony, tandis que la partie client est écrite en JavaScript et TypeScript.

User			
POST	/api/inscription	Creates a User resource.	⌵ 🔒
GET	/api/profileMe	Retrieves a User resource.	⌵ 🔒
GET	/api/user/{id}	Retrieves a User resource.	⌵ 🔒
DELETE	/api/userDelete/{id}	Removes the User resource.	⌵ 🔒
PATCH	/api/userProfileEdit/{id}	Updates the User resource.	⌵ 🔒
GET	/api/users	Retrieves the collection of User resources.	⌵ 🔒

Les Entités

Notre API est conçue pour servir notre application mobile Code Hub. Elle permet aux utilisateurs de saisir et stocker des informations dans trois entités distinctes : utilisateurs, articles et commentaires. Grâce à cette API, les données collectées peuvent être facilement traitées et utilisées pour alimenter les fonctionnalités de notre application mobile.



Entité user

```
#[ORM\Entity(repositoryClass: UserRepository::class)]
#[ORM\Table(name: 'user')]
#[ApiResource(
    operations: array(
        new GetCollection(
            uriTemplate: '/users',
            normalizationContext: ['groups' => 'users.read'],
            security: "is_granted('ROLE_ADMIN')"
        ),
        new Post(
            uriTemplate: '/inscription',
            denormalizationContext: array('groups' => 'user.write'),
            processor: UserProcessor::class
        ),
        new Get(
            uriTemplate: '/profileMe',
            normalizationContext: array('groups' => 'user.profile.me'),
            security: "is_granted('ROLE_ADMIN') or object == user",
            provider: UserStateProvider::class
        ),
    ),
)]
```

L'entité utilisateur (User) est définie en utilisant l'ORM Doctrine de Symfony. Elle est également exposée en tant que ressource API à travers ApiPlatform.

La ressource API "User" a plusieurs opérations définies, notamment "GetCollection" pour récupérer une collection d'utilisateurs, "Post" pour créer un nouvel utilisateur, "Get" pour récupérer un utilisateur spécifique et "Patch" pour mettre à jour un utilisateur spécifique.

Chaque opération dispose d'un "uriTemplate" qui définit l'URL correspondante pour accéder à l'opération, ainsi que d'autres contextes de normalisation, de dénormalisation et de sécurité qui sont appliqués lors de l'accès à l'opération.

State Provider:

Le State Provider est un concept clé dans API Platform qui permet de gérer l'état des ressources exposées par une API RESTful. En d'autres termes, le Stateprovider est responsable de :

- Gère la récupération des données dans API Platform
- Permet de modifier la récupération des données
- Permet de récupérer les données depuis un autre service et de les intégrer à API Platform

```
class CommentStateProvider implements ProviderInterface
{
    public function __construct(private readonly ArticleRepository $articleRepository, private Security $security)
    {}

    public function provide(Operation $operation, array $uriVariables = [], array $context = []): object|array|null
    {
        $comment = new Comment();
        $comment->setUserId($this->security->getUser());
        $article = $this->articleRepository->find($uriVariables['id']);
        $comment->setArticle($article);

        return $comment;
    }
}
```

L'image ci-dessus est un exemple dans notre API, CommentStateProvider est un Stateprovider personnalisé utilisé pour fournir une instance de la classe Comment lorsqu'une demande de création de ressource est effectuée.

Il récupère l'article correspondant à l'ID spécifié dans l'URI à partir d'un ArticleRepository, crée une nouvelle instance de la classe Comment, y ajoute l'utilisateur actuel à partir de la classe Security, et définit l'article récupéré.

La méthode provide renvoie ensuite cette nouvelle instance de Comment pour qu'elle soit persistée dans la source de données.

State Processor:

Le State Processor est un concept clé dans API Platform qui permet de manipuler les données de la ressource avant ou après son enregistrement en base de données, ou avant ou après l'affichage des données lorsqu'une demande est effectuée. Le State Processor est responsable de :

- Gère la persistance des données dans API Platform
- Permet de modifier la persistance des données
- Permet d'effectuer des opérations spécifiques lors de la persistance des données

```
class ArticleProcessor implements ProcessorInterface
{
    public function __construct(private Security $security, private readonly EntityManagerInterface $entityManager)
    {}

    public function process(mixed $data, Operation $operation, array $uriVariables = [], array $context = []): void
    {
        if(false === $data instanceof Article) {
            return;
        }
        $data->setUpdatedAt(new \DateTimeImmutable());
        if($operation->getName() == "_api_/articleEdit/{id}_patch"){
            $data->setCreatedAt($data->getCreatedAt());
        }
        elseif($operation->getName() == "_api_/articlePost_post"){
            $data->setUserId($this->security->getUser());
            $data->setCreatedAt(new \DateTimeImmutable());
        }
        else {
            $data->setCreatedAt(new \DateTimeImmutable());
        }

        $this->entityManager->persist($data);
        $this->entityManager->flush();
    }
}
```

L'image ci-dessus est un exemple dans notre API, ArticleProcessor est un State Processor utilisé pour manipuler les données de l'entité Article avant et après son enregistrement dans la base de données.

La méthode vérifie que les données sont bien une instance de la classe Article, puis modifie les propriétés updatedAt et createdAt en fonction de l'opération demandée.

Si l'opération est une mise à jour de l'article, la date de création ne sera pas modifiée. Si l'opération est une création de l'article, la date de création sera définie à la date et heure courantes, et l'utilisateur actuel sera également défini comme créateur de l'article.

Les tests unitaires :

Les tests unitaires sont des tests automatisés qui permettent de vérifier le bon fonctionnement d'une portion de code, souvent une méthode ou une classe, isolée du reste de l'application.

J'ai utilisé le framework PHPUnit et les Mocks qui sont des objets qui simulent le comportement d'un autre objet pour les besoins d'un test unitaire.

Dans le contexte d'API Platform, les tests unitaires peuvent être utilisés pour s'assurer du bon fonctionnement des classes comme les Processors et les Providers. Par exemple, un test unitaire pourrait vérifier si la méthode provide d'un StateProvider retourne bien les données attendues en fonction des paramètres d'entrée.

De même, un test unitaire pourrait vérifier si la méthode process d'un State Processor modifie correctement les données de la ressource en fonction de l'opération demandée.

```
public function setUp(): void
{
    $this->entityManager = $this->createMock(EntityManagerInterface::class);
    $this->user = $this->createMock(User::class);
    $this->userPasswordHasher = $this->createMock(UserPasswordHasherInterface::class);
    $this->operation = $this->createMock(Operation::class);
    $this->dateTimeImmutable = new DateTimeImmutable();
}

public function testProcess()
{
    $this->user->expects($this->once())->method('setUpdatedAt');
    $this->user->expects($this->once())->method('setCreatedAt');
    $this->user->expects($this->any())->method('getCreatedAt')->willReturn($this->dateTimeImmutable);

    $this->operation->expects($this->any())->method('getName')->willReturnOnConsecutiveCalls('_api_/articleEdit/{id}_patch');

    $this->user->expects($this->any())->method('getPlainPassword')->willReturn('password');
    $this->user->expects($this->once())->method('setPassword');
    $this->userPasswordHasher->expects($this->once())->method('hashPassword')->with($this->user, 'password');
    $this->user->expects($this->any())->method('getPassword');

    $processor = new UserProcessor($this->userPasswordHasher, $this->entityManager);

    $processor->process($this->user, $this->operation, $this->uriVariables, $this->context);
}
```

Les test fonctionnels :

Les tests fonctionnels, également appelés tests d'intégration, sont des tests automatisés qui permettent de vérifier le bon fonctionnement d'une application dans son ensemble, en testant plusieurs composants ensemble, tels que des API, des bases de données, des services tiers, etc.

Dans le contexte d'API Platform, les tests fonctionnels peuvent être utilisés pour tester l'API REST dans son ensemble, en vérifiant que les différentes ressources sont correctement exposées et que les opérations CRUD (Create, Read, Update, Delete) fonctionnent correctement.

Dans le cadre de notre API, nous avons utilisé le framework PHPUnit pour la mise en place de tests fonctionnels. Pour faciliter la rédaction de ces tests, nous avons créé une classe abstraite nommée "AbstractWebTestCase", qui contient les requêtes HTTP et les méthodes les plus fréquemment utilisées pour accéder à nos différents endpoints.

```
abstract class AbstractWebTestCase extends ApiTestCase
{
    private const USERS_INFO = [
        'ROLE_ADMIN' => ['email' => 'admin@test.com',
            'password' => 'password'],
        'ROLE_USER' => ['email' => 'user@test.com',
            'password' => 'password'],
    ];

    public function provideUsers(): Generator
    {
        yield 'ROLE_ADMIN' => [self::USERS_INFO['ROLE_ADMIN'], 'ROLE_ADMIN'];
        yield 'ROLE_USER' => [self::USERS_INFO['ROLE_USER'], 'ROLE_USER'];
    }

    public function getAdminToken(): string
    {
        $client = static::createClient();
        $jwtManager = $client->getContainer()->get(JWTTokenManagerInterface::class);
        $admin = static::getContainer()->get('doctrine')->getRepository(User::class)->findOneBy(['id' => 16]);

        $token = $jwtManager->create($admin);
        $this->assertIsString($token);

        return $token;
    }

    public function getUserToken(): string
    {
        $client = static::createClient();
    }
}
```

Nous avons créé des classes de test pour chaque entité de notre API, telles que "ArticleTest", "UserTest" et "CommentTest".

```
class UserTest extends AbstractWebTestCase
{
    /**
     * @dataProvider provideUsers
     * @throws TransportExceptionInterface
     * @throws ServerExceptionInterface
     * @throws RedirectionExceptionInterface
     * @throws ClientExceptionInterface
     */
    public function testAsUserOrAdminICanLogIn($user): void
    {
        $client = static::createClient();
        $response = $client->request('POST', '/api/login_check', ['json' => $user]);

        $this->assertEquals(200, $response->getStatusCode());
        $token = json_decode($response->getContent(), true);
        $this->assertArrayHasKey('token', $token);
        $this->assertIsString($token['token']);
    }

    /**
     * @throws RedirectionExceptionInterface
     * @throws ClientExceptionInterface
     * @throws TransportExceptionInterface
     * @throws ServerExceptionInterface
     * @throws \Exception
     */
    public function testAsAdminICanGetUsers(): void
    {

```

L'utilisation de cette stratégie nous a permis de réduire considérablement le temps et l'effort nécessaire pour écrire des tests pour notre API.

Nous avons pu réutiliser la classe "AbstractWebTestCase" dans plusieurs classes de tests et réduire la duplication de code, ce qui a également rendu nos tests plus cohérents et plus faciles à maintenir.

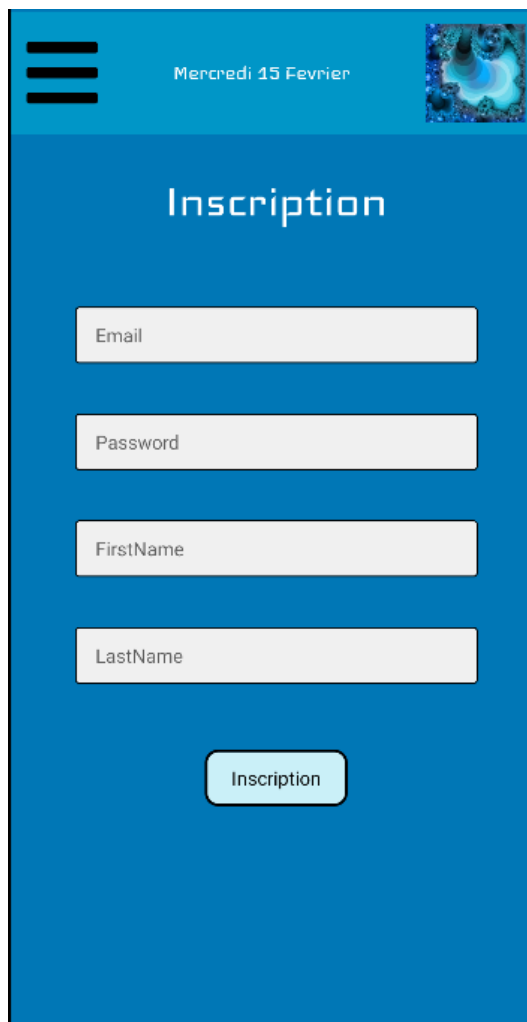
Les fonctionnalités du projet

Code hub est un forum en version application mobile. Ainsi, il regroupe les fonctionnalités classiques d'un forum que l'on pourrait retrouver sur un site web.

Utilisateur non connecté

Création d'un compte utilisateur

Un utilisateur non connecté peut créer un compte utilisateur. Pour ce faire, il doit renseigner un **courriel**, un nom et un prénom et définir un mot de passe.



The screenshot shows a mobile application interface for user registration. At the top, there is a blue header bar containing a hamburger menu icon on the left, the date "Mercredi 15 Fevrier" in the center, and a small profile picture on the right. Below the header, the word "Inscription" is displayed in a large, white, sans-serif font. Underneath, there are four white input fields with rounded corners, each containing a placeholder label: "Email", "Password", "FirstName", and "LastName". At the bottom of the form, there is a white button with rounded corners and a black border, labeled "Inscription". The entire form is set against a solid blue background.

Les articles

Les articles sont publics, c'est-à-dire qu'ils sont accessibles en lecture à toute personne ayant l'application disponible.

Un visiteur non identifié peut donc lire les articles librement.

Les utilisateurs connectés à l'application via leur compte peuvent consulter les articles eux aussi.

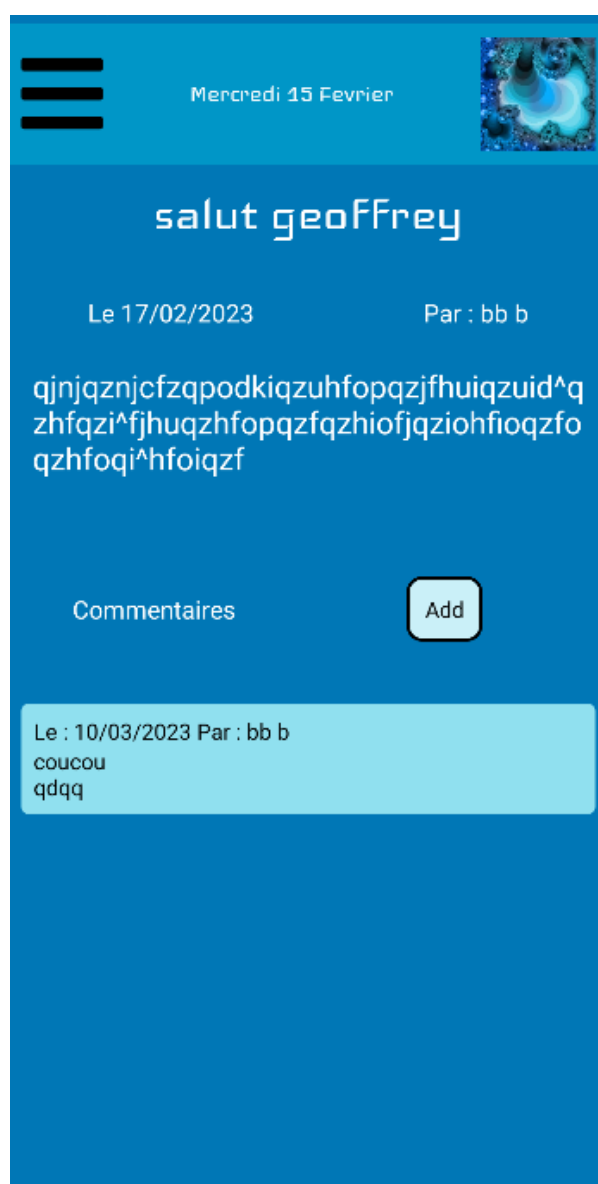


Les commentaires

Les commentaires sont faits pour enrichir un article.

Ainsi, leur lecture est libre pour toute personne ayant l'application Code hub disponible sur son téléphone et même s' il n'est pas identifié.

Les utilisateurs connectés à l'application via leur compte peuvent consulter les commentaires eux aussi.



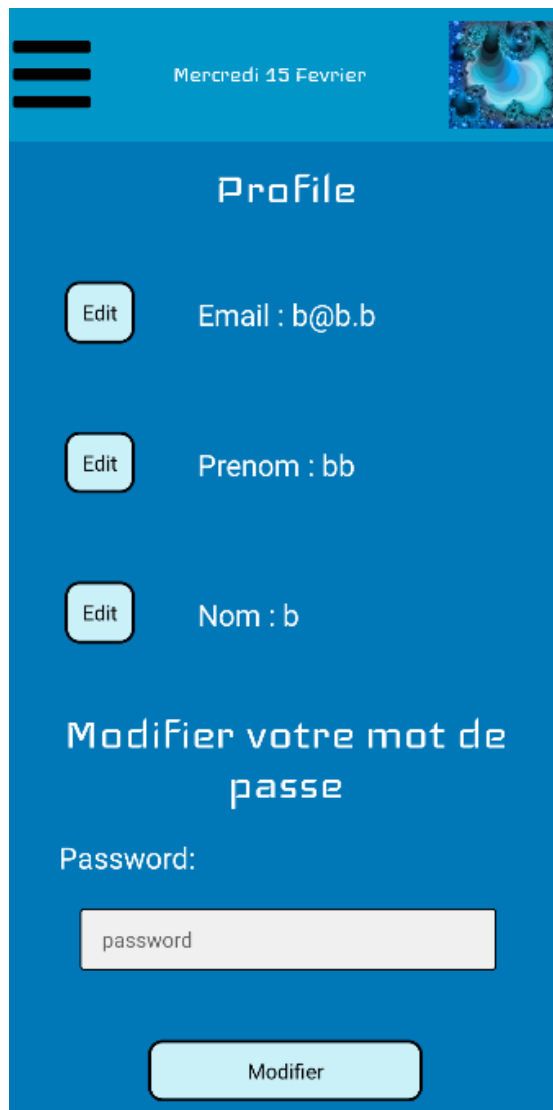
Utilisateur connecté

Administration de son compte utilisateur

Pour administrer son compte utilisateur il suffit de se rendre dans la section profil, disponible dans le menu.

La page permet de :

- ❖ Modifier son nom
- ❖ Modifier son prénom
- ❖ Modifier son mail
- ❖ Modifier son mot de passe
- ❖ Supprimer son compte



Mercredi 15 Fevrier

Profile

Edit Email : b@b.b

Edit Prenom : bb

Edit Nom : b

Modifier votre mot de passe

Password:

password

Modifier

Création d'articles

Créer un article se fait en se rendant sur la page "Création d'article" et l'on peut alors y rédiger un article d'un minimum de 750 caractères.

Une fois écrit, il suffit de l'enregistrer en actionnant le bouton "enregistrer" en bas de l'article.

Celui-ci est alors immédiatement publié et visible par l'ensemble des utilisateurs.



Mercredi 15 Février

Ajouter un article

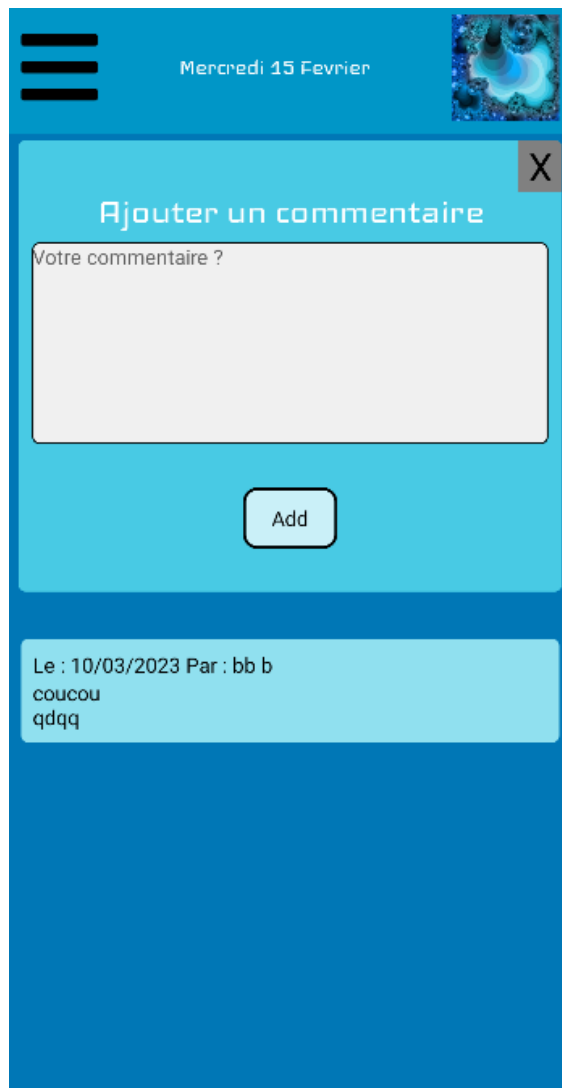
Titre de l'article :

Texte de votre article :

Création d'un commentaire

Pour créer un commentaire, il suffit d'aller sur l'article que l'on veut commenter et ajouter un commentaire via le bouton "ajouter un commentaire" en bas de l'article.

Une fois que l'on a rédigé le commentaire, il suffit de cliquer sur le bouton "enregistrer" pour que le commentaire soit visible et disponible sur l'application.



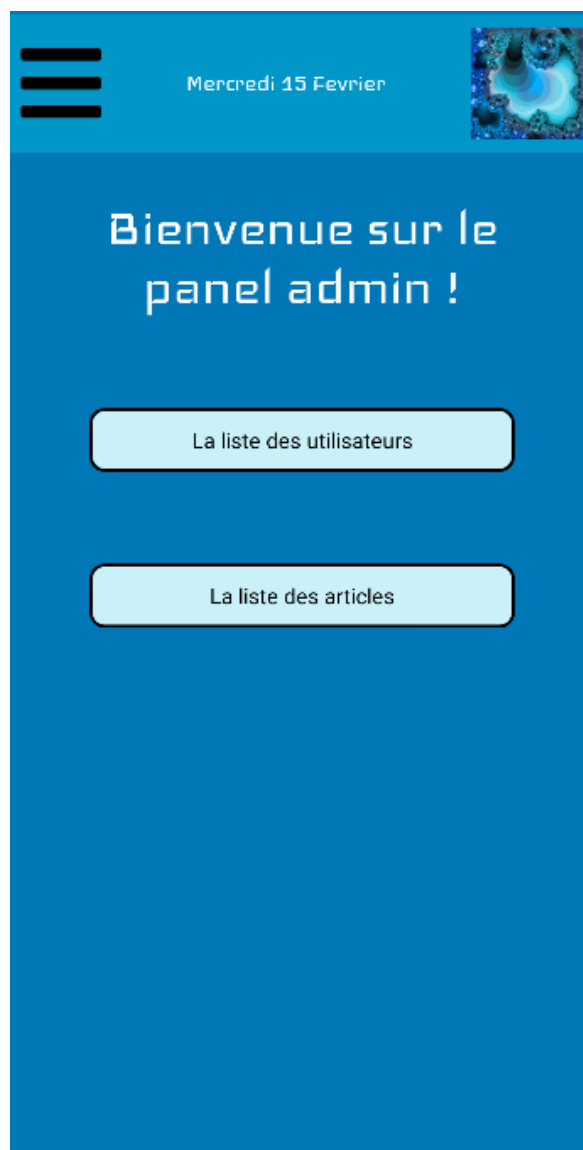
The screenshot shows a mobile application interface with a blue background. At the top, there is a header bar with a hamburger menu icon on the left, the date "Mercredi 15 Fevrier" in the center, and a small square image on the right. Below the header, there is a light blue box with the title "Ajouter un commentaire" and a close button (X) in the top right corner. Inside this box, there is a text input field with the placeholder text "Votre commentaire ?". Below the input field is a button labeled "Add". Below the light blue box, there is a white box containing the text "Le : 10/03/2023 Par : bb b", "coucou", and "qdpq".

Administrateur du site

L'administration du site se fait directement sur l'application.

Les utilisateurs avec un rôle administrateur peuvent accéder aux fonctionnalités décrites ci-dessous.

Nous avons voulu faire un panel administrateur intégré directement à l'application pour simplifier l'administration du site et proposer un produit 100% intégré et simple de déploiement.



Modération des articles

La modération d'article se fait via le menu "admin".

Cela ouvre une liste des articles paginés et triés dans leur ordre de rédaction.

Chaque article propose 2 choix, la suppression ou la modification.

Pour modérer un article, il faut prendre l'option modification.

L'ouverture de l'article permet de le modifier, puis de l'enregistrer à nouveau. L'article est alors modifié.



The image shows a web form titled "Modification de l'article n° 401" on a blue background. The form contains the following elements:

- Titre de l'article :** A text input field containing "salut geoffrey".
- Contenue de l'article :** A large text area containing a long string of random characters: "qjnjqznjcfzqpodkiqzuhfopqzjfhuizuid*qzhfqzi*fjhuqzhfopqzfqzhiofjqziohfioqzfoqzhfoqi*hfioqzf".
- Buttons:** Two buttons labeled "Modifier" and "Annuler".
- Commentaires:** A section header followed by "N° 96" and a text input field containing "coucou qdqq".

Suppression des articles

La suppression d'article se fait simplement en cliquant sur la poubelle.

Un pop-up vous demande de confirmer la demande de suppression, une fois valider. L'article est totalement supprimé.

Liste des articles

Article n° 401 :

salut geoffrey

qjnjqznjcfzqpodkiqzuhfopqzjfhuizuid^
qzhfqzi^fjhuqzhfopqzfqzhiofjqziohfioq
zfoqzhfoqi^hfoiqzf

ModifierSupprimer

Modération des commentaires

La modération des commentaires se fait sur la même page que la modération des articles. Chaque commentaire est accessible pour pouvoir être modifié. Il suffit de modifier l'article directement dans le champ texte et cliquer sur "enregistrer" le commentaire.

Suppression des commentaires

La suppression des commentaires se fait en cliquant sur le bouton "Supprimer" du commentaire que l'on veut supprimer.

Back-end

Ajout et appelle d'une route

Pour ajouter et appeler une route dans le back-end, il vous suffit de vous rendre sur le fichier api.js qui se situe à la racine du projet

```
export const getArticleById = (id, callback) => {
  request("get", `/article/${id}`, null, (res) => {
    return callback(res)
  });
}
export const postArticle = (title, content, callback) => {
  request("post", `/articlePost`, { title, content }, (res) => {
    return callback(res)
  });
}
export const patchArticle = (id, title, content, callback) => {
  request("patch", `/articleEdit/${id}`, { title, content }, (res) => {
    return callback(res)
  });
}
export const deleteArticle = (id, callback) => {
  request("delete", `/articleDelete/${id}`, null, (res) => {
    return callback(res)
  });
}
```

Nous utilisons axios pour effectuer une requête vers l'API, et une méthode request a été créée qui gère les réponses des requêtes.

```
const request = async (method, url, data, callback) => {
  await getJwtToken();
  header = { 'Content-type': 'application/json', }
  if (method == "patch") {
    header = { 'Content-type': 'application/merge-patch+json' }
  }

  return api({
    method: method,
    url: url,
    data: data,
    headers: header
  }).then(res => {
    return callback(res);
  })
  .catch(error => {
    console.log(error.message)
    console.log(`Erreur ${error.response.data.code}`)
    console.log(error.response.data.message)
    if (error.response.data.message == 'Expired JWT Token') {
      SecureStore.deleteItemAsync('jwt').then(() => {
        Alert.alert('Votre session a expiré', 'Veuillez vous re-connecter pour continuer', [
          { text: 'OK', onPress: () => { } }
        ])
        return null
      })
    }
    return callback(error.response)
  });
};
```

La fonction request ne requiert que 3 paramètres, le premier consiste à définir la méthode (get/post/patch/delete), le second la route, le troisième les paramètres de la requête si il y en a, et enfin elle renvoie en callback la réponse (ou l'erreur).

Pour ensuite appeler votre requête dans n'importe quel fichier, il suffit d'importer le fichier api.js ainsi que la méthode que vous avez définies

```
import { getArticles } from '../api';

export default function IndexArticleScreen({ navigation }) {
  const route = useRoute();
  const refresh = route.params.refresh;
  console.log(refresh)

  const [articles, setArticles] = useState([]);
  const [loading, setLoading] = useState(false);
  const [page, setPage] = useState(1);
  const [totalItems, setTotalItems] = useState(0);

  const fetchData = () => {
    setLoading(true);
    getArticles(page, (res) => {
      setArticles(prevArticles => [...prevArticles, ...res.data['hydra:member']]);
      setTotalItems(res.data['hydra:totalItems']);
      setLoading(false);
    });
  };
};
```

Si paramètre il y a, il suffit de les envoyer à la méthode en premier temps, et ensuite récupérer la réponse en fonction fléchée anonymes.

Projet d'avenir et prochaines fonctionnalités

Je vois dans les prochaines fonctionnalités à prévoir dans codehub, divers éléments :

- Le nombre de vues des articles.
- Classement des articles par nombre de vues pour promouvoir les contenus les plus populaires.
- Ajouter des images d'illustrations dans les articles.

Conclusion

Le projet CodeHub repose sur plusieurs objectifs et a été développé par une équipe de quatre développeurs dans le cadre de notre formation en tant que "Concepteurs Développeurs d'Applications".

Ce projet représente le point culminant de notre formation et nous a permis d'acquérir une expérience variée dans la mise en place, la conception et la création d'une application.

Code Hub propose un forum permettant aux utilisateurs de publier des articles et de les commenter. Cette fonctionnalité favorise les échanges et les discussions au sein de la communauté.

L'application dispose également d'une interface dédiée aux administrateurs, qui leur permet de gérer les utilisateurs, de modérer, de modifier ou de supprimer les articles, ainsi que de faire de même avec les commentaires.

Cette fonctionnalité donne aux administrateurs un contrôle complet sur le contenu de l'application et garantit un environnement sûr et de qualité pour les utilisateurs.

En résumé, le projet Code Hub est le fruit de notre formation en tant que développeur d'applications.

Il nous a permis d'acquérir une expérience précieuse dans la mise en place, la conception et la création d'une application complète.

Avec des fonctionnalités telles qu'un forum d'articles et une interface d'administration, Code Hub vise à offrir une expérience interactive et enrichissante pour les utilisateurs et les administrateurs.