# Documentation Technique mobile

# Sommaire

# connexion

```
ElevatedButton(
  onPressed: () {
    if (_formKey.currentState!.validate()) {
      String email = _emailController.text;
      String password = _passwordController.text;
      _login(email, password);
    }
  },
  child: Text('Se connecter'),
```

```
class _LoginFormState extends State<LoginForm> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  TextEditingController _emailController = TextEditingController();
  TextEditingController _passwordController = TextEditingController();
```

Le composant LoginForm est responsable de l'affichage du formulaire de connexion et de la gestion de la tentative de connexion de l'utilisateur.

_login: C'est une fonction définie dans la classe _LoginFormState qui est appelée lorsque le bouton est pressé. Cette fonction est responsable de la gestion de la tentative de connexion de l'utilisateur.
Paramètres de _login: La fonction _login prend deux paramètres en entrée : email et password, qui représentent respectivement l'e-mail et le mot de passe saisis par l'utilisateur dans les champs du formulaire.

# Connexion

- Corps de la fonction _login: À l'intérieur de la fonction _login, un appel HTTP est effectué vers une URL spécifique du serveur (http:/api/usersroute/connexionAdmin) pour vérifier les informations d'identification de l'utilisateur.
- Envoi des données au serveur: Les données d'identification de l'utilisateur (e-mail et mot de passe) sont envoyées au serveur sous forme de corps de requête JSON à l'aide de la fonction http.post. Les données sont encodées en JSON à l'aide de jsonEncode.

```
void _login(String email, String password) async {
  try {
    String apiUrl = 'http://localhost:3000/api/usersroute/connexionAdmin';
    Map<String, String> data = {
      'email': email,
      'password': password,
    };
    final response = await http.post(
      Uri.parse(apiUrl),
      body: jsonEncode(data),
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
      },
    );

    print('Server Response: ${response.statusCode}');
    print('Server Data: ${response.body}');

    if (response.statusCode == 200) {
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => DashboardPage()),
```

# connexion

- Extraction des données: Elle extrait l'e-mail et le mot de passe à partir du corps de la requête HTTP.
- Connexion à la base de données: Elle établit une connexion à la base de données.
- Requête SQL: Elle effectue une requête SQL pour trouver un utilisateur correspondant à l'e-mail fourni. Si aucun utilisateur n'est trouvé ou si l'utilisateur n'est pas administrateur, une erreur 401 est renvoyée.
- Vérification du mot de passe: Si un utilisateur administrateur est trouvé, le mot de passe fourni est vérifié par rapport à celui stocké dans la base de données à l'aide de la bibliothèque bcrypt.

```
exports.connexionAdmin = async (req, res) => {
    const { email, password } = req.body;
    try {
        const connection = await db.getConnection();
        const [result] = await connection.query(
            'SELECT * FROM users WHERE email = ? AND admin = 1',
            [email]
        );
        connection.release();
        if (result.length === 0) {
            return res.status(401).json({ error: 'Accès refusé. Vous n\'êtes pas administrateur.' });
        }
        const user = result[0];
        const passwordMatch = await bcrypt.compare(password, user.password);
        if (!passwordMatch) {
            return res.status(401).json({ error: 'Mot de passe incorrect' });
        }
        const token = jwt.sign({
            uid: user.uid,
            email: user.email,
            isAdmin: true
        }, process.env.API_KEY, { expiresIn: '1h' });
        res.json({ token });
    } catch (error) {
        console.error('Erreur lors de la connexion admin :', error);
        res.status(500).json({ error: 'Une erreur de connexion' });
    }
};
```

```
), // ElevatedButton
const SizedBox(height: 16),
ElevatedButton(
  onPressed: () {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => AddProductPage(),
```

# Navigation

- La navigation entre les différentes pages est gérée à l'aide du widget Navigator de Flutter.
- Lorsqu'un bouton est pressé, une nouvelle route est ajoutée à la pile de navigation, affichant ainsi la page correspondante.

```
class AddProductPage extends StatefulWidget {
  const AddProductPage({super.key});

  @override
  _AddProductPageState createState() => _AddProductPageState();
}
```

# Page magasin

Dépendances et Importations :
http: Pour effectuer des requêtes HTTP vers l'API du magasin.
cached_network_image: Pour afficher les images des produits avec mise en cache.
material.dart: Pour les widgets de l'interface utilisateur Material Design.

```dart
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:cached_network_image/cached_network_image.dart';

class Product {
  final int id;
  final String name;
  final String details;
  final double price;
  final String image;
  final int quantity; // Christian Latour
```

Chaque produit est caractérisé par les attributs suivants :
id: L'identifiant unique du produit.
name: Le nom du produit.
details: Les détails ou la description du produit.
quantity: La quantité disponible du produit.
price: Le prix du produit.
image: Le chemin de l'image du produit.

# Page magasin

Récupération des Données

Les données des produits sont récupérées à partir d'une API distante en utilisant la fonction fetchProducts(). Cette fonction envoie une requête HTTP GET à l'URL de l'API du magasin pour obtenir la liste des produits disponibles.

```
Future<void> fetchProducts() async {
  final response = await http
      .get(Uri.parse('http://localhost:3000/api/produitsroute/produit'));
  if (response.statusCode == 200) {
    List<dynamic> data = jsonDecode(response.body);
    setState(() {
      products = data.map((item) => Product.fromJson(item)).toList();
    });
  } else {
    throw Exception('Failed to load products');
  }//christian latour
}
```

Affichage des Produits

Les produits sont affichés dans un ListView.builder sur la page principale de l'application. Chaque élément de la liste représente un produit et affiche son nom, ses détails, sa quantité, son prix et son image associée. L'image du produit est récupérée à partir de l'URL fournie par l'API.

```
child: ListView.builder(
  itemCount: products.length,
  itemBuilder: (context, index) {
    return Padding(
      padding: const EdgeInsets.all(8.0),
      child: Card(
        elevation: 4,
        child: ListTile(
          contentPadding: const EdgeInsets.all(8.0),
          title: Text(
            products[index].name,
            style: const TextStyle(
              fontWeight: FontWeight.bold,
              fontSize: 18.0,
            ), // TextStyle
          ), // Text
          subtitle: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text(
                products[index].details,
                style: const TextStyle(fontSize: 16.0),
              ), // Text
              Text(
                'Prix: ${products[index].price.toStringAsFixed(2)}', // Affichage du prix
                style: const TextStyle(
                  fontSize: 16.0, fontWeight: FontWeight.bold), // TextStyle
              ), // Text
              Text(
                'Quantité: ${products[index].quantity}', // Affichage de la quantité
                style: const TextStyle(fontSize: 16.0),
              ), // Text
            ],
          ), // Column
          trailing: SizedBox(
            width: 100,
            child: AspectRatio(
              aspectRatio: 1,
              child: CachedNetworkImage(
                imageUrl:
                    'http://localhost:3000/${products[index].image}',
                placeholder: (context, url) =>
                    const CircularProgressIndicator(),
                errorWidget: (context, url, error) => const Icon(Icons.error),
                fit: BoxFit.cover,//christian latour
```

# Ajouter un produit

- Dépendances et Importations
- La page d'ajout de produit utilise plusieurs packages Flutter importés pour des fonctionnalités spécifiques :
- http: Pour effectuer des requêtes HTTP vers l'API du magasin.
- image_picker: Pour sélectionner une image à partir de la galerie du périphérique.
- material.dart: Pour les widgets de l'interface utilisateur Material Design

```dart
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:cached_network_image/cached_network_image.dart';
```

Les informations saisies par l'utilisateur sont encapsulées dans un modèle de données Product, qui comprend les attributs suivants :
name: Le nom du produit.
details: Les détails ou la description du produit.
price: Le prix du produit.
quantity: La quantité disponible du produit.
image: Le chemin de l'image du produit.

```dart
'Name: ${_nameController.text}');
'details: ${_detailsController.text}');
'Price: ${_priceController.text}');
'Quantity: ${_quantityController.text}');
'Image path: $_imagePath');
```

# Ajouter un produit

- Envoi des Données
- Une fois que l'utilisateur a saisi les informations du produit et sélectionné une image, il peut soumettre le formulaire. Les données du produit, y compris l'image sous forme de fichier, sont envoyées à l'API du magasin via une requête HTTP POST a l’adresse api/produitsroute/produit'

```dart
Future<void> _submitProduct() async {
  if (_nameController.text.isEmpty ||
      _detailsController.text.isEmpty ||
      _priceController.text.isEmpty ||
      _quantityController.text.isEmpty ||
      _image == null) {
    print('Veuillez remplir tous les champs.');
    return;
  }
  print('Name: ${_nameController.text}');
  print('details: ${_detailsController.text}');
  print('Price: ${_priceController.text}');
  print('Quantity: ${_quantityController.text}');
  print('Image path: $_imagePath');
  try {
    final url = Uri.parse('http://localhost:3000/api/produitsroute/produit');
    List<int> imageBytes = await _image!.readAsBytes();
    final imageUploadRequest = http.MultipartRequest('POST', url);
    final imagePart = http.MultipartFile.fromBytes(
      'image',
      imageBytes,
      filename: 'product_image.jpg', //
    );
    imageUploadRequest.files.add(imagePart);
    imageUploadRequest.fields['name'] = _nameController.text;
    imageUploadRequest.fields['details'] = _detailsController.text;
    imageUploadRequest.fields['price'] = _priceController.text;
    imageUploadRequest.fields['quantity'] = _quantityController.text;
    final streamedResponse = await imageUploadRequest.send();
    final response = await http.Response.fromStream(streamedResponse);
    if (response.statusCode == 200) {
      print('Produit ajouté avec succès.');
    } else {
      print('Erreur lors de l\'ajout du produit: ${response.reasonPhrase}');
    }
  } catch (error) {
    print('Erreur lors de lajout du produit: $error');
  }
}//christian latour
```

# Modifier un produit

Dépendances et Importations
La page de modification de produit utilise plusieurs packages Flutter importés pour des fonctionnalités spécifiques :

```dart
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:cached_network_image/cached_network_image.dart';
```

http: Pour effectuer des requêtes HTTP vers l'API du magasin.
cached_network_image: Pour afficher les images des produits à partir de leur URL.
material.dart: Pour les widgets de l'interface utilisateur Material Design.

```dart
Future<void> _fetchProducts() async {
  try {
    final response = await http.get(
        Uri.parse('http://localhost:3000/api/produitsroute/produit'));
    if (response.statusCode == 200) {
      List<dynamic> data = jsonDecode(response.body);
      setState(() {
        _products = data.map((item) => Product.fromJson(item)).toList();
      });
    } else {
      throw Exception('Failed to load products');
    }
  } catch (error) {
    print('Error fetching products: $error');
  }
}//christian latour
```

Les données des produits sont récupérées depuis l'API du magasin à l'aide de requêtes HTTP GET. Les produits sont ensuite affichés dans une liste, où chaque élément de la liste représente un produit.

# Modifier un produit

Soumission de la Modification au Serveur
Une requête HTTP de type PUT est envoyée au serveur avec l'URL lié au pid du produit
/api/produitsroute/produit/$pid

Affichage de la Boîte de Dialogue de Modification

```dart
void _showEditForm(Product product) {
  TextEditingController nameController =
      TextEditingController(text: product.name);
  TextEditingController detailsController =
      TextEditingController(text: product.details);
  TextEditingController priceController =
      TextEditingController(text: product.price.toString());
  TextEditingController quantityController =
      TextEditingController(text: product.quantity.toString());
//christian latour
```

```dart
void _submitProductModification(String pid, String newName,
    String newdetails, double newPrice, int newQuantity) async {
  try {
    final updatedProduct = Product(
      pid: pid,
      name: newName,
      details: newdetails,
      price: newPrice,
      image: '',
      quantity: newQuantity,
    );
    final response = await http.put(
      Uri.parse('http://localhost:3000/api/produitsroute/produit/$pid'),
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
      },
      body: jsonEncode(updatedProduct.toJson()),
    );
    if (response.statusCode == 200) {
      print('Produit modifié avec succès.');
    } else {
      print('Erreur lors de la modification du produit: ${response.reasonPhrase}');
    }
  } catch (error) {
    print('Erreur lors de la modification du produit: $error');
  }//christian latour
}
```

# Supprimer un produit

- Pour supprimer un produit, l'utilisateur peut appuyer sur l'icône de suppression à droite de chaque élément de la liste. Cela déclenche une requête HTTP DELETE vers l'API pour supprimer le produit de la base de données.

```dart
void _deleteProduct(String pid) async {
  try {
    final response = await http.delete(
      Uri.parse('http://localhost:3000/api/produitsroute/produit/$pid'),
    );
    if (response.statusCode == 200) {
      print('Produit supprimé avec succès.');
      setState(() {
        _products.removeWhere((product) => product.pid == pid);
      });
    } else {
      print('Erreur lors de la suppression du produit: ${response.reasonPhrase}');
    }
  } catch (error) {
    print('Erreur lors de la suppression du produit: $error');
  }
}//christian latour
```

# Déconnexion

- Pour la déconnexion, une redirection est effectuée en remplaçant la pile de routes par la page de connexion, ce qui empêche l'utilisateur de revenir en arrière avec le bouton de retour.

```
void _logout(BuildContext context) {
  Navigator.pushReplacement(
    context,
    MaterialPageRoute(
      builder: (context) =>
          MyApp(),
    ), // MaterialPageRoute
  );
}
```

```
actions: <Widget>[
  IconButton(
    onPressed: () => _logout(context),
    icon: const Row(
      children: [
        Icon(Icons.exit_to_app),
        SizedBox(width: 8),
        Text(
          'Déconnexion',
          style: TextStyle(fontSize: 16),
```