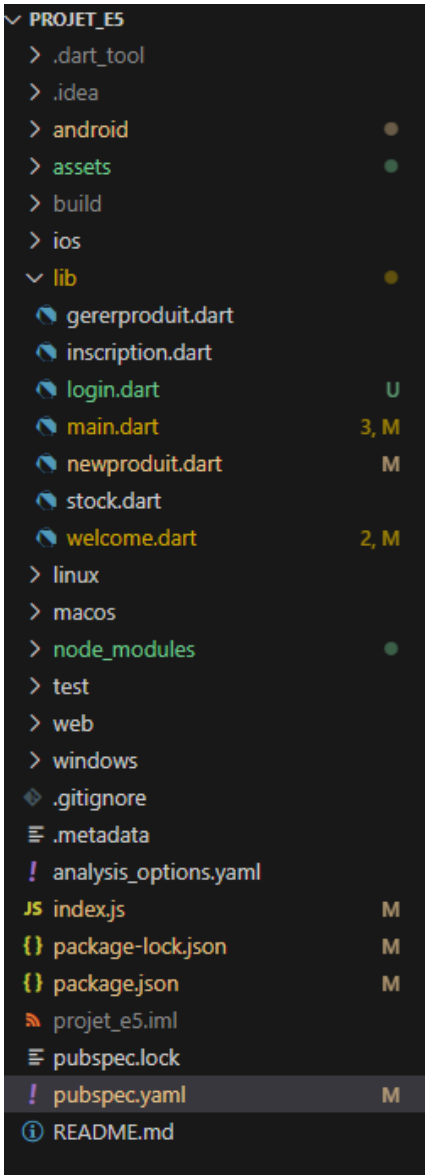
Application Mobile (E6) Persona World – en cours

Nom : Mustapha Chouhani
Option : BTS SIO SLAM
Réalisation : Application mobile Flutter + API Node.js + MySQL
Année scolaire : 2024-2025

1. Contexte

L'application mobile Persona World est une interface destinée aux administrateurs d'un espace marchand. Elle permet actuellement de créer un compte, de se connecter, et prévoit à terme la gestion des produits ainsi que des commandes. Le développement est encore en cours, mais les fonctionnalités d'inscription, de connexion sécurisée, et l'intégration avec une base de données sont déjà opérationnelles.

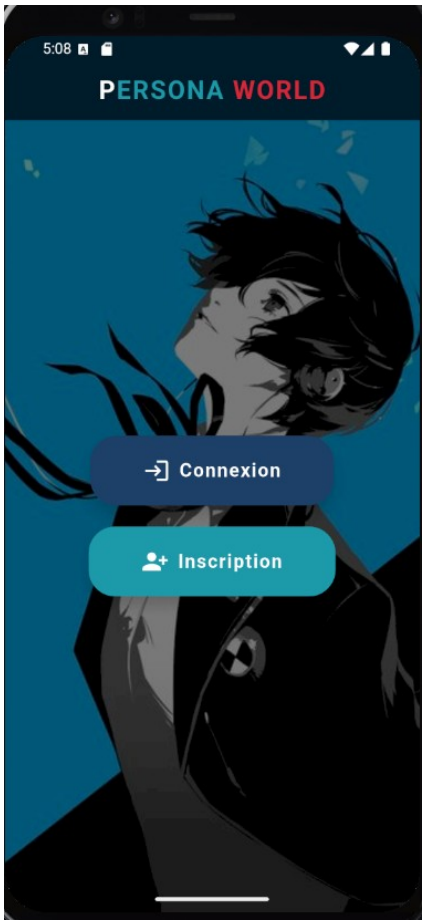


2. Structure du projet

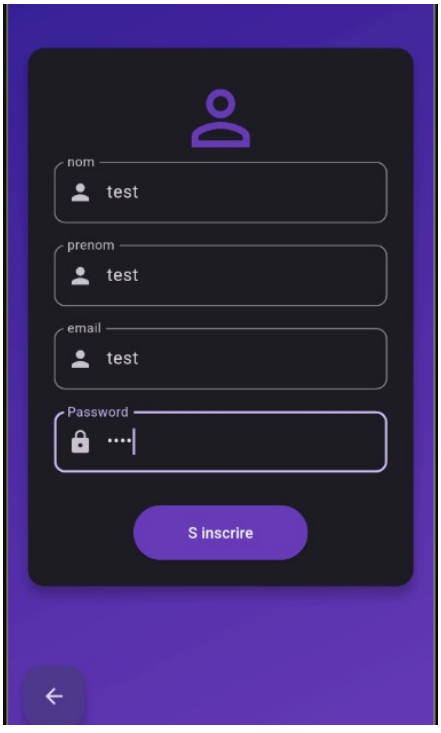
Le projet est structuré autour de Flutter pour l'interface mobile, d'une API développée en Node.js, et d'une base de données MySQL. L'application envoie des requêtes HTTP pour communiquer avec le serveur. Les données utilisateur sont stockées de manière sécurisée, avec hashage des mots de passe via bcrypt.

Structure du projet dans Visual Studio Code avec les fichiers front-end (Flutter) et backend (Node.js).

3. Fonctionnalités implémentées



Écran d'accueil de l'application : permet de choisir entre inscription et connexion.



Formulaire d'inscription avec les champs nom, prénom, email et mot de passe.

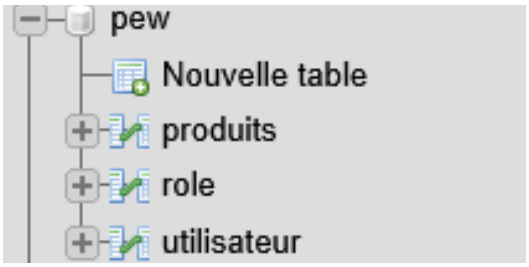
Code Flutter – Requête HTTP POST vers l'API pour l'enregistrement.

```
Windsurf: Refactor | Explain | Generate Function Comment | X
Future<void> envoyerDemande() async {
  const String apiUrl = "http://10.0.2.2:3000/user";

  // Envoi de la requête POST
  final response = await http.post(
    Uri.parse(apiUrl),
    headers: {"Content-Type": "application/json"},
    body: jsonEncode({
      "nom": nomController.text,
      "prenom": prenomController.text,
      "email": emailController.text,
      "password": passwordController.text,
    }),
  );

  // Affichage des détails de la réponse dans la console
  print("Réponse de l'API : ${response.statusCode}");
  if (response.statusCode == 200) {
    // Success
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text("Demande envoyée avec succès.")),
    );

    nomController.clear();
    prenomController.clear();
    emailController.clear();
    passwordController.clear();
  } else {
    // Affiche les erreurs éventuelles dans la console
    print("Erreur de l'API : ${response.body}");
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text("Erreur lors de l'envoi de la demande.")),
    );
  }
}
```

Code Node.js – Traitement de l'inscription, hashage du mot de passe, insertion en base.

```
app.post('/user', async  (req, res) => {
  const { nom, prenom, email, password } = req.body;

  try {
    const hashedPassword = await bcrypt.hash(password, 10);

    const sql = 'INSERT INTO `utilisateur` (nom, prenom, email, password) VALUES (?, ?, ?, ?)';
    db.query(sql, [nom, prenom, email, hashedPassword], (err, result) => {
      if (err) {
        return res.status(500).send(err);
      }
      res.json({
        id: result.insertId,
        nom,
        prenom,
        email
      });
    });
  } catch (err) {
    res.status(500).send('Erreur lors du hashage du mot de passe');
  }
});
```

Structure de la base MySQL, table 'utilisateur' avec les colonnes nécessaires.



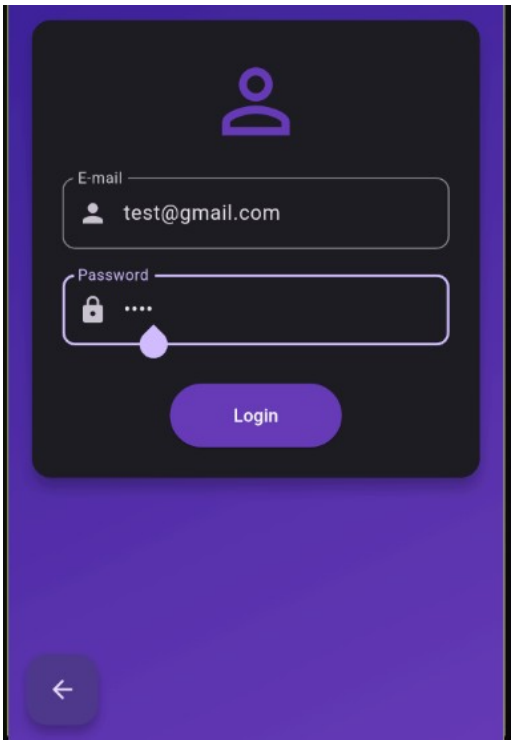
Aperçu de la structure de la table 'utilisateur' dans phpMyAdmin.

1	**id**	int		Non	*Aucun(e)*	AUTO_INCREMENT
2	**email**	varchar(100)	utf8mb4_0900_ai_ci	Non	*Aucun(e)*	
3	**password**	varchar(256)	utf8mb4_0900_ai_ci	Non	*Aucun(e)*	
4	**nom**	varchar(100)	utf8mb4_0900_ai_ci	Non	*Aucun(e)*	
5	**prenom**	varchar(100)	utf8mb4_0900_ai_ci	Non	*Aucun(e)*	

Exemple de données enregistrées en base avec mot de passe hashé.

	id	email	password	nom	prenom
Éditer Copier Supprimer	5	test@gmail.com	$2b$10$uWx3tAlPjhw/eXJr2z3NE.Bsy1jAzhXA85OZOBNw9eF...	test	test

Formulaire de connexion utilisateur avec champs email et mot de passe.

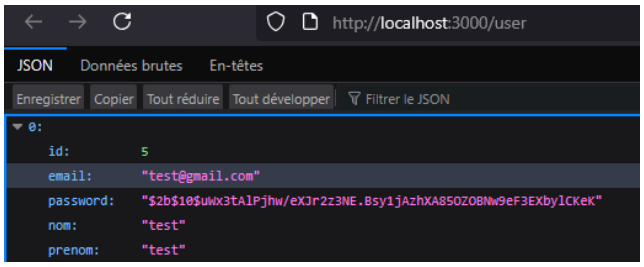


```dart
class _LoginPageState extends State<LoginPage> {
  final _formKey = GlobalKey<FormState>();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();

  Windsurf: Refactor | Explain | Generate Function Comment | X
  Future<void> _login() async {
    final email = _emailController.text;
    final password = _passwordController.text;

    if (email.isEmpty || password.isEmpty) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Veuillez remplir tous les champs')),
      );
      return;
    }

    final url = Uri.parse('http://10.0.2.2:3000/login');

    try {
      final response = await http.post(
        url,
        headers: {'Content-Type': 'application/json'},
        body: json.encode({
          'email': email,
          'password': password,
        }),
      );

      if (response.statusCode == 200) {
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text('Connexion réussie')),
        );
```

Code Flutter – Requête HTTP POST vers l'API de connexion.

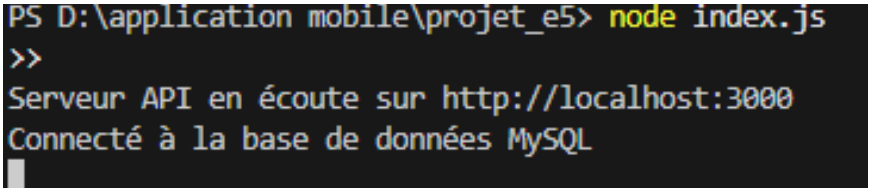
```node
app.post('/login', async (req, res) => {
  const { email, password } = req.body;

  if (!email || !password) {
    return res.status(400).json({ message: '12345689'});
  }
  const sql = 'SELECT * FROM `utilisateur` WHERE email = ?';
  db.query(sql, [email], async (err, results) => {
    if (err) {
      return res.status(500).send(err);
    }
    if (results.length === 0) {
      return res.status(401).json({ message: 'Emailt de passe incorrect' });
    }

    const utilisateur = results[0];

    bcrypt.compare(password, utilisateur.password, (err, result) => {
      if (err) {
        return res.status(500).send(err);
      }
      if (!result) {
        return res.status(401).json({ message:'email ou mdp incorrect'});
      }

      res.json({ message: 'Connexion reussite', utilisateur });
    });
  });
});
```

Code Node.js – Vérification des identifiants et réponse API.



Affichage du retour JSON de l'API après connexion.

Interface du tableau de bord avec accès à la gestion des produits et commandes.



Serveur Node.js en écoute, prêt à recevoir les requêtes.