Cahier des charge ayant pour objectif l'étude et l'accompagnement dans la réalisation de l'Atelier Professionnel n°4 –M2L.

AP4 – M2L

Documentation technique

Florent BERNIER – BTS SIO SLAM

Table of Contents

API.dart - addProduct

```dart
Future<void> addProduct(File? image, Map<String, dynamic> productData) async {
  String addProductUrl = '$url/boutique/additem';

  try {
    String? authToken = UserData.userDataConnected?['token'];
    if (authToken == null) {
      throw Exception('Utilisateur non connecté');
    }

    String? imageName = image != null ?
'img-${DateTime.now().millisecondsSinceEpoch}${path.extension(image.path)}' : null;

    var request = http.MultipartRequest('POST', Uri.parse(addProductUrl));

    if (image != null) {
      var imageStream = http.ByteStream(image.openRead());
      var imageLength = await image.length();
      var imageMultipart = http.MultipartFile(
        'thumbnail',
        imageStream,
        imageLength,
        filename: imageName,
      );
      request.files.add(imageMultipart);
    }

    Map<String, String> productDataString = productData.map((key, value) => MapEntry(key,
value.toString()));

    Map<String, String> imageStringData = image != null ? {'thumbnail': imageName!} : {};

    request.fields.addAll({...productDataString, ...imageStringData});

    request.headers['Content-Type'] = 'multipart/form-data';
    request.headers['Authorization'] = authToken;

    var response = await request.send();

    if (response.statusCode == 200) {
      print('Produit ajouté avec succès');
    } else {
      throw Exception(
          'Erreur lors de l\'ajout du produit souhaité: ${response.reasonPhrase}\n');
    }
  } catch (e) {
    throw Exception('Erreur lors de l\'ajout du produit : $e\n');
  }
}
```

Expression des besoins

Pour ce 4e Atelier Professionnel, nous allons devoir réaliser une application mobile à l'aide de Flutter et Dart..

Ce projet a pour objectif de développer nos compétences à travailler sur un langage différent et de s'organiser.

L'application mobile permettra aux employés (staff et administrateurs) d'accéder à la liste des produits et des utilisateurs, dans laquelle il va pouvoir : supprimer, modifier et ajouter un produit, et ajouter, modifier et supprimer un utilisateur.

Les rôles obligatoires

- **Administrateur (2)** : Il aura la possibilité d'ajouter, modifier ou supprimer un utilisateur, un staff, un produit, et a l'accès à la base de donnée.
- **Staff (1)** : Il aura la possibilité d'ajouter, modifier ou supprimer un produit, et a la possibilité de contrôler les utilisateurs ou staff.
- **Utilisateur (0)** : Il a aucun accès à l'application.

Détails de la base de donnée

Table utilisateurs

La table « users » est utilisé pour stocké les informations de l'utilisateur :

- L'ID de l'utilisateur, mis en « AUTO_INCREMENT » qui augmentera à chaque nouvel utilisateur ajouté.
- Le prénom et nom de famille, qui sont le prénom et le nom de l'utilisateur, mis par défaut sur « NULL », n'étant pas obligatoire pour l'utilisateur.
- Le pseudonyme et l'email, étant tout deux utilisés pour la connexion à son compte utilisateur, et son unique évitant la création de plusieurs comptes sur la même adresse mail, et l'utilisation de plusieurs fois le même nom d'utilisateur.
- Le mot de passe est le texte caché, permettant à l'utilisateur de se connecter à son compte, devant être haché pour éviter l'usurpation d'identité.
- Le pays, permettant à l'utilisateur d'indiquer son pays résident, qui sera majoritairement « France » dans le cas du projet.
- Le niveau de permission, qui par défaut est défini à 0. Ce paramètre indique le pouvoir que le compte utilisateur possède :
    - Niveau 0 : Utilisateur
    - Niveau 1 : Staff
    - Niveau 2 : Administrateur

#	Nom	Type de données	Taille/Ensem...	Non signé	NULL autorisé	ZEROFILL	Par défaut
1	id	INT	10	☐	☐	☐	AUTO_INCREME...
2	firstname	VARCHAR	40	☐	☑	☐	NULL
3	lastname	VARCHAR	40	☐	☑	☐	NULL
4	username	VARCHAR	40	☐	☐	☐	Pas de défaut
5	email	VARCHAR	40	☐	☐	☐	Pas de défaut
6	pwd	VARCHAR	250	☐	☐	☐	Pas de défaut
7	country	VARCHAR	40	☐	☑	☐	NULL
8	perm_level	INT	11	☐	☐	☐	'0'

Table category_item

La table « category_item » permet de stocker les différentes catégories d'item :

- L'ID, mis en « AUTO_INCREMENT », qui définit l'id de la catégorie.
- Le nom de la catégorie (par exemple : ballon, vêtement, équipements, ... etc.).

#	Nom	Type de données	Taille/Ensem...	Non signé	NULL autorisé	ZEROFILL	Par défaut
1	id	INT	10	☐	☐	☐	AUTO_INCREME...
2	name	VARCHAR	40	☐	☐	☐	''

Table items

La table « items » permet de stocker les informations d'un article :

- L'ID, mis en « AUTO_INCREMENT », qui définit l'id de l'article.
- Le nom de l'article, qui sera affiché sur le site.
- La description de l'article, affichée sur le site donnant des informations sur l'article.
- Les stocks, permettant d'afficher le nombre d'article restant.
- L'image de l'article, affichant à quoi ressemble l'article.
- Le prix de l'article, indiquant à l'utilisateur le prix de l'article.

#	Nom	Type de données	Taille/Ensem...	Non signé	NULL autorisé	ZEROFILL	Par défaut
1	id_items	INT	10	☐	☐	☐	AUTO_INCREME...
2	item_name	VARCHAR	40	☐	☐	☐	Pas de défaut
3	description	LONGTEXT		☐	☐	☐	Pas de défaut
4	stocks	INT	10	☐	☐	☐	Pas de défaut
5	thumbnail	LONGTEXT		☐	☑	☐	NULL
6	price	FLOAT		☐	☐	☐	Pas de défaut
7	id_category	INT	11	☐	☑	☐	NULL

Table panier

La table « cart » permet d'afficher le panier d'un utilisateur :

- L'ID.
- Le nom du produit.

- La quantité d'article ajoutés.
- Le prix total des articles dans le panier.
- L'id_item, faisant référence à l'article ajouté dans le panier.
- L'id_user, étant l'identifiant de l'utilisateur ayant ajouté le produit.

#	Nom	Type de données	Taille/Ensem...	Non signé	NULL autorisé	ZEROFILL	Par défaut
1	id	INT	11	☐	☐	☐	AUTO_INCREME...
2	item_name	VARCHAR	50	☐	☐	☐	Pas de défaut
3	quantity	INT	11	☐	☐	☐	Pas de défaut
4	item_price	FLOAT		☐	☐	☐	Pas de défaut
5	id_item	INT	11	☐	☐	☐	Pas de défaut
6	id_user	INT	11	☐	☐	☐	Pas de défaut

Ressources et code

Connexion à la BDD

Pour recevoir les informations de la base de donnée sur l'application, nous passons par le lien du serveur où nous pourrons effectuer les requêtes déjà préparées.

Page API.dart

La page API.dart est la page où les requêtes de l'application sont effectuées. On récupère sur cette page via l'url du serveur, les routes pour l'application mobile.

API – Connexion de l'utilisateur

API.dart - authenticateUser

```dart
String url = 'http://192.168.1.28:8001';

class UserData {
  static Map<String, dynamic>? userDataConnected;
}

Future<Map<String, dynamic>> authenticateUser(
    String email, String password) async {
  String loginUrl = '$url/users/login';

  Map<String, String> body = {
    'email': email,
    'password': password,
  };

  Map<String, String> headers = {
    'Content-Type': 'application/json',
  };

  try {
    http.Response response = await http.post(
      Uri.parse(loginUrl),
      headers: headers,
      body: jsonEncode(body),
    );
    debugPrint('connexion: ${response.body}');

    if (response.statusCode == 200) {
      Map<String, dynamic> userDataConnected = jsonDecode(response.body);
      debugPrint('token : ${userDataConnected['token']}');
      debugPrint('userData: ${userDataConnected['user']}');

      UserData.userDataConnected = userDataConnected;

      int permLevel = userDataConnected['user']['permLevel'] as int;
      if (permLevel == 1 || permLevel == 2) {
        return userDataConnected;
      } else {
        throw Exception(
            'Permission insuffisante pour accéder à l\'application');
      }
    } else if (response.statusCode == 404) {
      throw Exception('Utilisateur non trouvé');
    } else {
      throw Exception('Mot de passe incorrect');
    }
  } catch (e) {
    debugPrint('Erreur lors de l\'authentification : $e');
    throw Exception('Erreur lors de l\'authentification : $e');
  }
}
```

API – Récupérer la liste des utilisateurs

API.dart - fetchUsers

```dart
Future<List<dynamic>> fetchUsers() async {
  String usersUrl = '$url/users';

  try {
    String? authToken = UserData.userDataConnected?['token'];
    if (authToken == null) {
      throw Exception('Token invalide ou expiré');
    }

    Map<String, String> headers = {
      'Content-Type': 'application/json',
      'Authorization': authToken,
    };

    final response = await http.get(Uri.parse(usersUrl), headers: headers);

    if (response.statusCode == 200) {
      return List<dynamic>.from(jsonDecode(response.body));
    } else if (response.statusCode == 401) {
      throw Exception('Utilisateur non connecté');
    } else {
      throw Exception('Failed to load users: ${response.statusCode}');
    }
  } catch (e) {
    throw Exception('Erreur lors du chargement des utilisateurs : $e');
  }
}
```

API – Récupérer la liste des produits

API.dart - fetchItems

```dart
Future<List<dynamic>> fetchItems() async {
  String itemUrl = '$url/boutique';

  try {
    String? authToken = UserData.userDataConnected?['token'];
    if (authToken == null) {
      throw Exception('Token invalide ou expiré');
    }

    Map<String, String> headers = {
      'Content-Type': 'application/json',
      'Authorization': authToken,
    };

    http.Response response =
    await http.get(Uri.parse(itemUrl), headers: headers);

    if (response.statusCode == 200) {
      return List<dynamic>.from(jsonDecode(response.body));
    } else {
      throw Exception('Failed to load items: ${response.statusCode}');
    }
  } catch (e) {
    throw Exception('Erreur lors du chargement des articles : $e');
  }
}
```

API – Ajouter un utilisateur employé

API.dart - addUser

```dart
Future<void> addUser(Map<String, dynamic> userData) async {
  String addEmployeeUrl = '$url/users/addemployee';

  try {
    String? authToken = UserData.userDataConnected?['token'];
    if (authToken == null) {
      throw Exception('Utilisateur non connecté');
    }

    Map<String, String> headers = {
      'Content-Type': 'application/json',
      'Authorization': authToken,
    };

    final response = await http.post(
      Uri.parse(addEmployeeUrl),
      headers: headers,
      body: jsonEncode(userData),
    );

    if (response.statusCode == 200) {
      print('Utilisateur ajouté avec succès');
    } else {
      throw Exception('Erreur lors de l\'ajout de l\'utilisateur: ${response.statusCode}');
    }
  } catch (e) {
    throw Exception('Erreur lors de l\'ajout de l\'utilisateur : $e');
  }
}
```

API – Ajouter un produit

API.dart - addProduct

```dart
Future<void> addProduct(File? image, Map<String, dynamic> productData) async {
  String addProductUrl = '$url/boutique/additem';

  try {
    String? authToken = UserData.userDataConnected?['token'];
    if (authToken == null) {
      throw Exception('Utilisateur non connecté');
    }

    String? imageName = image != null ?
'img-${DateTime.now().millisecondsSinceEpoch}${path.extension(image.path)}' : null;

    var request = http.MultipartRequest('POST', Uri.parse(addProductUrl));

    if (image != null) {
      var imageStream = http.ByteStream(image.openRead());
      var imageLength = await image.length();
      var imageMultipart = http.MultipartFile(
        'thumbnail',
        imageStream,
        imageLength,
        filename: imageName,
      );
      request.files.add(imageMultipart);
    }

    Map<String, String> productDataString = productData.map((key, value) => MapEntry(key,
value.toString()));

    Map<String, String> imageStringData = image != null ? {'thumbnail': imageName!} : {};

    request.fields.addAll({...productDataString, ...imageStringData});

    request.headers['Content-Type'] = 'multipart/form-data';
    request.headers['Authorization'] = authToken;

    var response = await request.send();

    if (response.statusCode == 200) {
      print('Produit ajouté avec succès');
    } else {
      throw Exception(
          'Erreur lors de l\'ajout du produit souhaité: ${response.reasonPhrase}\n');
    }
  } catch (e) {
    throw Exception('Erreur lors de l\'ajout du produit : $e\n');
  }
}
```

API – Supprimer un utilisateur

API.dart - deleteUser

```dart
Future<void> deleteUser(int userId) async {
  String deleteUserUrl = '$url/users/deleteUser/$userId';

  try {
    String? authToken = UserData.userDataConnected?['token'];
    if (authToken == null) {
      throw Exception('Utilisateur non connecté');
    }

    Map<String, String> headers = {
      'Content-Type': 'application/json',
      'Authorization': authToken,
    };

    final response = await http.delete(
      Uri.parse(deleteUserUrl),
      headers: headers,
    );

    if (response.statusCode == 200) {
      print('Utilisateur supprimé avec succès');
    } else {
      throw Exception('Erreur lors de la suppression de l\'utilisateur: ${response.statusCode}');
    }
  } catch (e) {
    throw Exception('Erreur lors de la suppression de l\'utilisateur : $e');
  }
}
```

API – Supprimer un produit

API.dart - deleteProduct

```dart
Future<void> deleteProduct(int productId) async {
  String deleteProductUrl = '$url/boutique/deleteitem/$productId';

  try {
    String? authToken = UserData.userDataConnected?['token'];
    if (authToken == null) {
      throw Exception('Utilisateur non connecté');
    }

    Map<String, String> headers = {
      'Content-Type': 'application/json',
      'Authorization': authToken,
    };

    final response = await http.delete(
      Uri.parse(deleteProductUrl),
      headers: headers,
    );

    if (response.statusCode == 200) {
      print('Produit supprimé avec succès');
    } else {
      throw Exception('Erreur lors de la suppression du produit: ${response.statusCode}');
    }
  } catch (e) {
    throw Exception('Erreur lors de la suppression du produit : $e');
  }
}
```

API – Mettre à jour un utilisateur

API.dart - updateUser

```dart
Future<void> updateUser(int userId, Map<String, dynamic> userData) async {
  String updateUserUrl = '$url/users/updateUser/$userId';

  try {
    String? authToken = UserData.userDataConnected?['token'];
    if (authToken == null) {
      throw Exception('Utilisateur non connecté');
    }

    Map<String, String> headers = {
      'Content-Type': 'application/json',
      'Authorization': authToken,
    };

    final response = await http.put(
      Uri.parse(updateUserUrl),
      headers: headers,
      body: jsonEncode(userData),
    );

    if (response.statusCode == 200) {
      print('Utilisateur mis à jour avec succès');
    } else {
      throw Exception('Erreur lors de la mise à jour de l\'utilisateur: ${response.statusCode}');
    }
  } catch (e) {
    throw Exception('Erreur lors de la mise à jour de l\'utilisateur : $e');
  }
}
```

API – Mettre à jour un produit

API.dart - updateProduct

```dart
Future<void> updateProduct(int productId, Map<String, dynamic> productData) async {
  String updateProductUrl = '$url/boutique/updateitem/$productId';
  try {
    String? authToken = UserData.userDataConnected?['token'];

    if (authToken == null) {
      throw Exception('Utilisateur non connecté');
    }

    Map<String, String> headers = {
      'Content-Type': 'application/json',
      'Authorization': authToken,
    };

    final response = await http.put(
      Uri.parse(updateProductUrl),
      headers: headers,
      body: jsonEncode(productData),
    );

    if (response.statusCode == 200) {
      // Le produit a été mis à jour avec succès
      print('Produit mis à jour avec succès');
    } else {
      // Une erreur s'est produite lors de la mise à jour du produit
      throw Exception('Erreur lors de la mise à jour du produit: ${response.statusCode}');
    }
  } catch (e) {
    throw Exception('Erreur lors de la mise à jour du produit : $e');
  }
}
```