SMOGON FORUM

TABLE DES MATIÈRES

LISTE DES COMPÉTENCES :

1. Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité :

- Maquetter une application
- Développer des composants d'accès aux données
- Développer la partie front-end d'une interface utilisateur web
- Développer la partie back-end d'une interface utilisateur web
- Développer une interface utilisateur de type desktop(non couvert)

2. Concevoir et développer la persistance des données en intégrant les recommandations de sécurité :

- Concevoir une base de données
- Mettre en place une base de données
- Développer des composants dans le langage d'une base de données

3. Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité :

- Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement (à discuter)
- Concevoir une application
- Développer des composants métier
- Construire une application organisée en couches
- Développer une application mobile
- Préparer et exécuter les plans de tests d'une application
- Préparer et exécuter le déploiement d'une application

Résumé du projet en anglais :

The project is a mobile application built using the MAUI framework for front-end development, supported by .NET 6.0 and a MySQL database. It serves as an adaptation of the popular online forum, Smogon Forums, which focuses on Pokémon-related discussions, team building, and e-sports competitions. The objective of the project is to create a mobile app that replicates the functionalities of the forum while incorporating additional features.

One of the key features of the app is the ability to build Pokémon teams using the website https://pokepast.es/. Users can leverage this feature to create their own teams and save the corresponding link on their profile page within the app. This allows for easy access to their preferred team compositions.

Furthermore, the application includes a dedicated tab that displays the website https://pokemonshowdown.com/. This integration enables users to directly access and play Pokémon battles on Pokémon Showdown using their saved teams or even explore and utilize teams shared by other members of the forum.

By bringing together the functionalities of the forum, team building capabilities, and integration with Pokémon Showdown, the mobile app aims to provide a comprehensive platform for Pokémon enthusiasts to engage in discussions, refine their team strategies, and actively participate in Pokémon battles.

Overall, the project offers a convenient and user-friendly mobile experience for accessing the Smogon Forums, facilitating team building, and enhancing the Pokémon gaming experience through seamless integration with external resources.

CAHIER DES CHARGES :

A. Présentation de l’entreprise :

1. Objectifs de l’application
2. Cibles
3. Type d’application
4. Équipement de vos cibles
5. Périmètre du projet

B. Graphisme et ergonomie :

1. Charte graphique
2. Wireframe
3. Maquettage

C. Spécificités et livrables :

1. Contenu de votre application
2. Contraintes techniques
3. Planning

A. Présentation de l'entreprise :

Extrait du site :

" *Smogon is the most comprehensive and accurate online resource for competitive Pokémon battling. We offer articles and advice via our community forums to help fans of the game compete at every level, while honing their skills in every aspect of competitive Pokémon from team building to battling tactics. Our over-450,000-member organization is growing at an ever increasing rate, constantly expanding our knowledge base and our ability to be at the cutting edge of the game.*

Though officially established December 18th, 2004 by the site owner, chaos, the history of our organization stretches back many years. Some members have been part of the online competitive Pokémon scene since 1999, originating from websites such as GameFAQs, Azure Heights, RPGamer, Pojo, and The PokeMasters. Top battlers from these websites held a number of tournaments, hosted on the original "Pokémon Battle Simulator," a web based sim written by Blizzard for the RBY Pokémon games. Other battle simulators were made, more strategies were developed, and a close knit community began to form around the game. This community needed a unifying presence on the Internet, a definitive home. And thus, Smogon began.

Since that time, we have evolved from just a tiny group of gaming gurus into a gigantic association of Pokémon fans whose membership spans the globe. The impressive growth of Smogon has brought it quite a bit of attention and recognition. We are now widely acknowledged on the web as being the authority in the competitive Pokémon arena.

Smogon also provides an outlet in the form of tournaments for competitive battlers to apply what they have learned from the vast resources provided on this site. At any given time there may be over fifteen active tournaments, varying in theme and structure, almost all of which are open to any registered member who would like to participate. Respected community members are always attempting to come up with some new twist to put into these exciting competitions that keep them fresh for all members. [...] "

Analyse de l'existant :

L'association SMOGON University possède un site internet responsive complet et fonctionnel sans aucune forme de monétisation.

A. 1. Les objectifs de l'application :

Quantitatifs :

- Apporter de la visibilité supplémentaire au site web existant
- Faire grandir la communauté de SMOGON

Qualitatifs :

- Produire une application du même standing que le site
- Permettre une maintenabilité et une accessibilité du code accru pour les membres développeur et futurs développeur de la communauté SMOGON

A. 2. Les cibles :

L'application a pour cible les passionnés des jeux pokémons, tout autant que les néophytes qui trouvent, grâce à SMOGON, une porte d'entrée sur la scène E-Sport pokémon.

A. 3. Le type d'application :

L'application sera un forum qui permet de poster des équipes pokémon pour partage simple ou pour demander des critiques. Ces équipes seront exploitables sur de nombreux supports grâce au format texte pokemon

A. 4. L'équipement de vos cibles :

L'application sera android et IOS, mobile uniquement (le site internet est déjà très optimisé pour les tablettes)

A. 5. Périmètre du projet :

- L'application a pour vocation de regrouper la communauté internationale qui utilise principalement l'anglais comme vecteur de communication, et sera ainsi intégralement en anglais.
- L'application intégrera la pokeapi (https://pokeapi.co/) pour toutes les données en matière de pokémon.
- Aucune monétisation n'est prévue ou à prévoir.
- Les spécificités que le prestataire doit connaître sont : la création de compte, les notifications push et l'exploitation du formatage équipe.

B. Graphisme et ergonomie :

B. 1. La charte graphique :

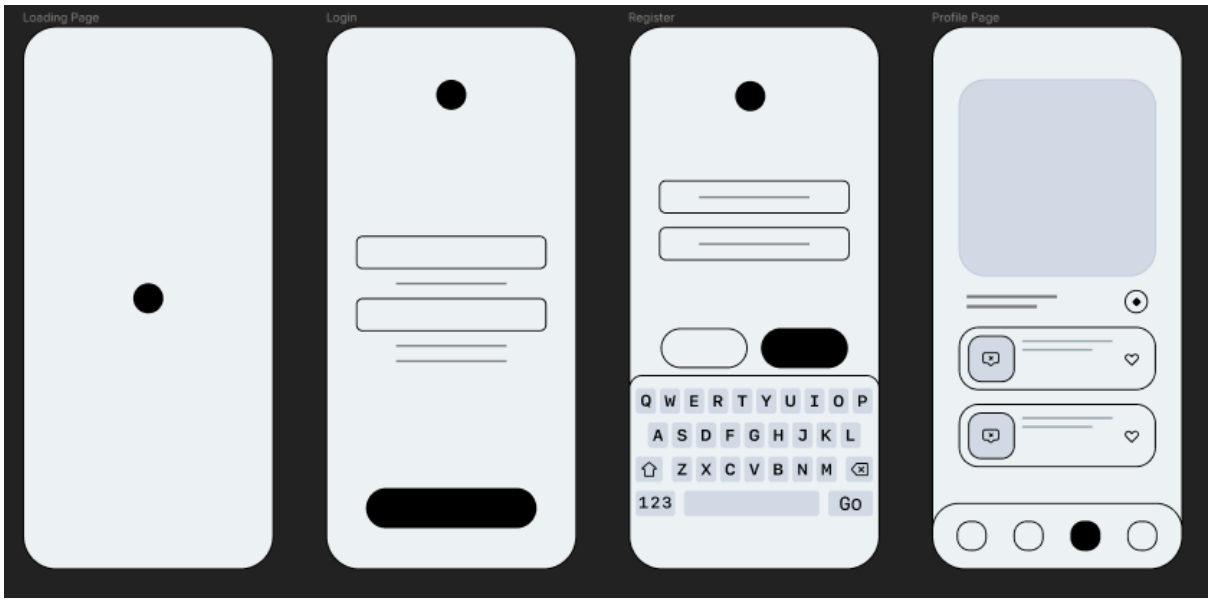
#ECECEC #6363B0 #8d8dcd

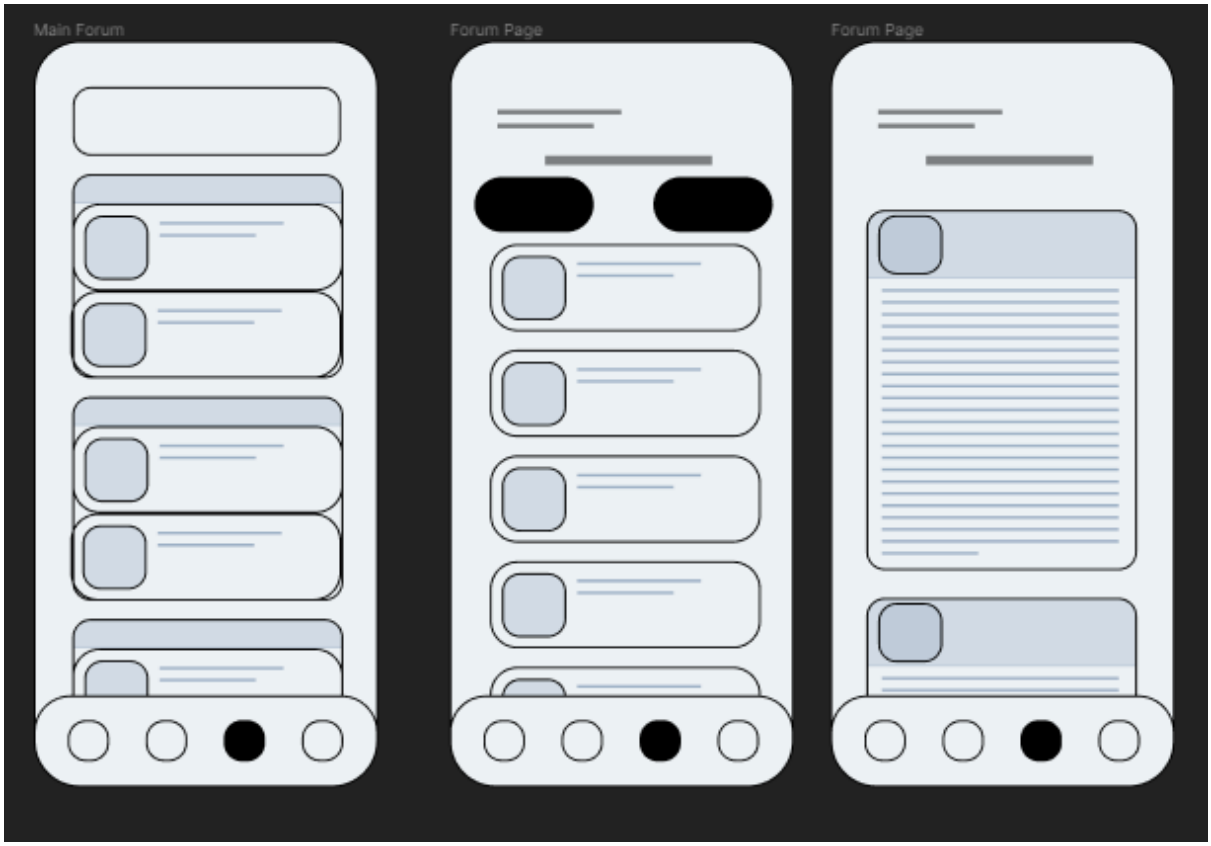
Main Font : Roboto Regular #000000

Exemples, counts, etc : Robot Regular #8C8C8C

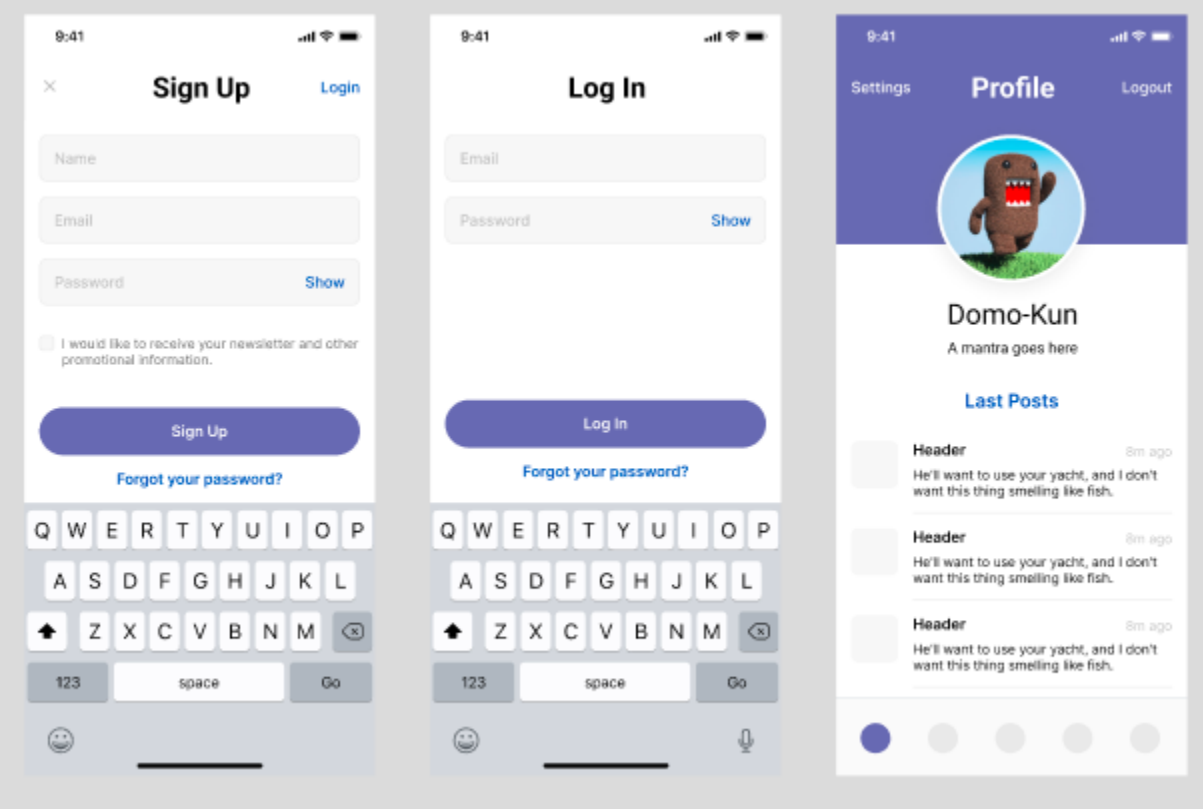
Important Text : Roboto Bold #094ABA

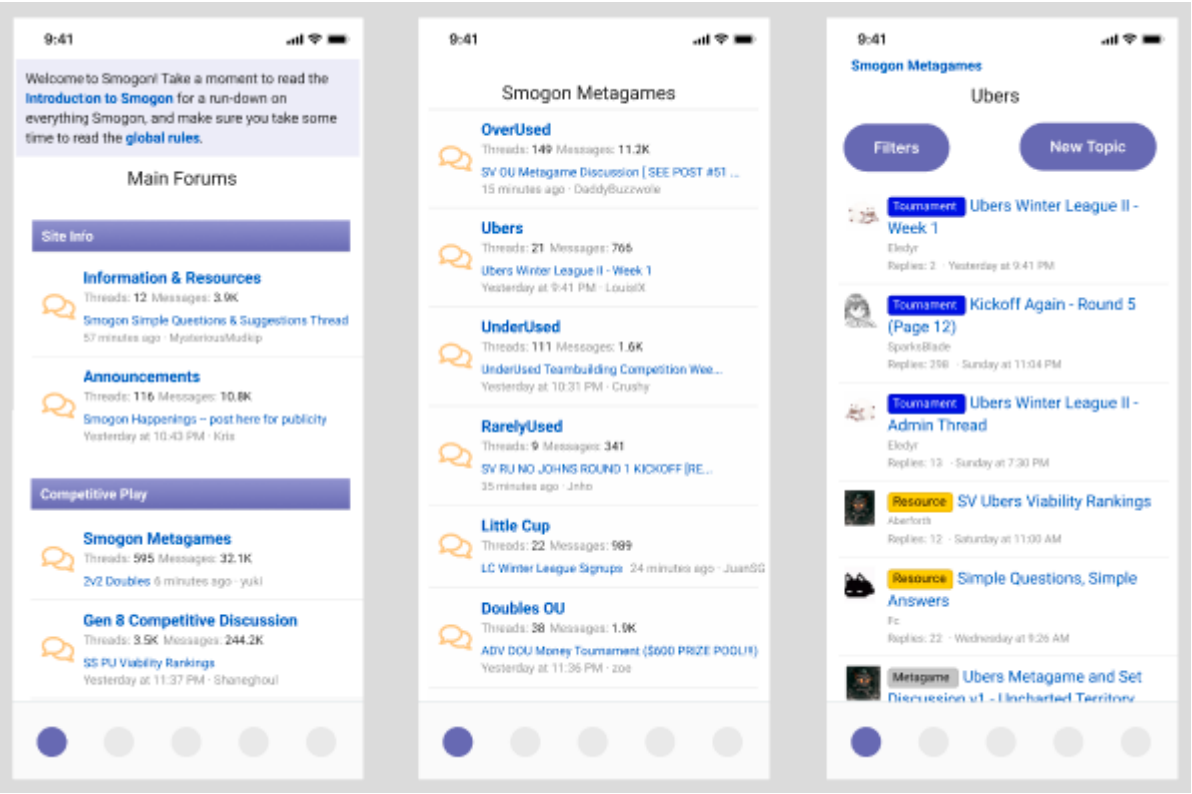
B. 2. WIREFRAME :

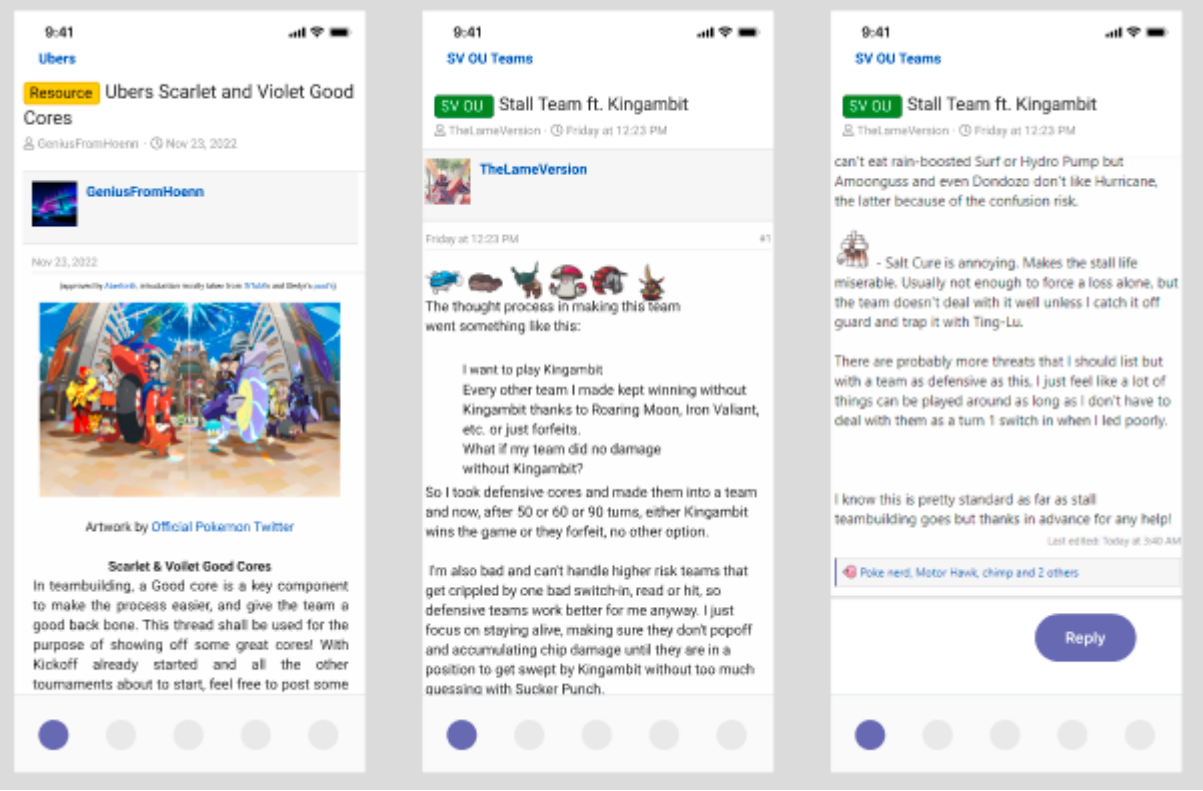


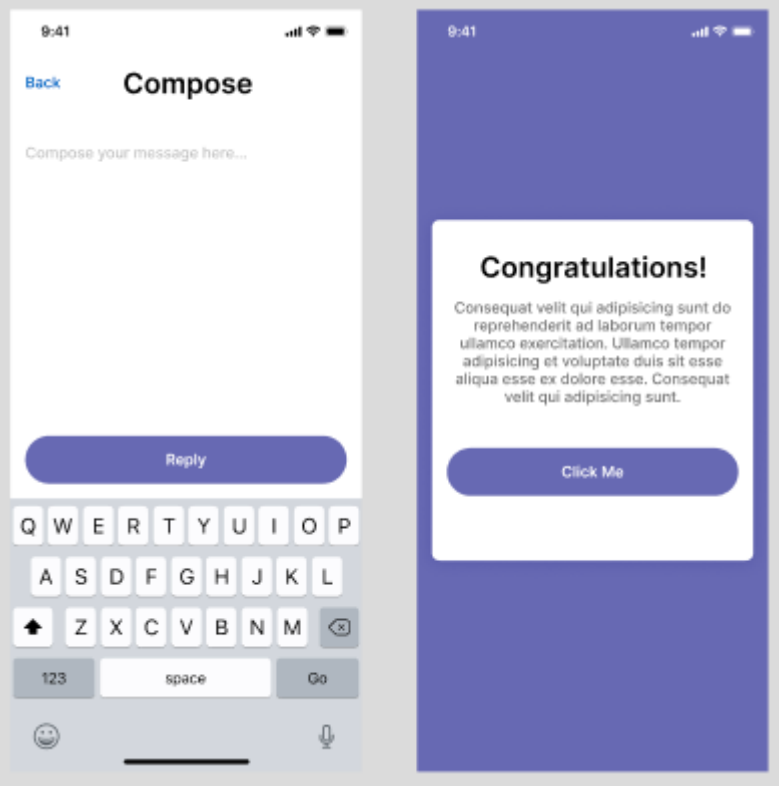


B. 3. MAQUETTAGE :






9:41
Ubers
Resource Ubers Scarlet and Violet Good Cores
GeniusFromHoenn · Nov 23, 2022
GeniusFromHoenn
Nov 23, 2022
Artwork by Official Pokemon Twitter
Scarlet & Voilet Good Cores
In teambuilding, a Good core is a key component to make the process easier, and give the team a good back bone. This thread shall be used for the purpose of showing off some great cores! With Kickoff already started and all the other tournaments about to start, feel free to post some
9:41
SV OU Teams
SV OU Stall Team ft. Kingambit
TheLameVersion · Friday at 12:23 PM
TheLameVersion
Friday at 12:23 PM
#1
The thought process in making this team went something like this:
I want to play Kingambit
Every other team I made kept winning without Kingambit thanks to Roaring Moon, Iron Valiant, etc. or just forfeits.
What if my team did no damage without Kingambit?
So I took defensive cores and made them into a team and now, after 50 or 60 or 90 turns, either Kingambit wins the game or they forfeit, no other option.
I'm also bad and can't handle higher risk teams that get crippled by one bad switch-in, read or hit, so defensive teams work better for me anyway. I just focus on staying alive, making sure they don't popoff and accumulating chip damage until they are in a position to get swept by Kingambit without too much guessing with Sucker Punch.
9:41
SV OU Teams
SV OU Stall Team ft. Kingambit
TheLameVersion · Friday at 12:23 PM
can't eat rain-boosted Surf or Hydro Pump but Amoonguss and even Dondozo don't like Hurricane, the latter because of the confusion risk.
- Salt Cure is annoying. Makes the stall life miserable. Usually not enough to force a loss alone, but the team doesn't deal with it well unless I catch it off guard and trap it with Ting-Lu.
There are probably more threats that I should list but with a team as defensive as this, I just feel like a lot of things can be played around as long as I don't have to deal with them as a turn 1 switch in when I led poorly.
I know this is pretty standard as far as stall teambuilding goes but thanks in advance for any help!
Last edited: Today at 3:40 AM
Poke nerd, Motor Hawk, chimp and 2 others
Reply


9:41
Back
Compose
Compose your message here...
Reply
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
Go
9:41
Congratulations!
Consequat velit qui adipisicing sunt do reprehenderit ad laborum tempor ullamco exercitation. Ullamco tempor adipisicing et voluptate duis sit esse aliqua esse ex dolore esse. Consequat velit qui adipisicing sunt.
Click Me

C. SPÉCIFICITÉS ET LIVRABLES :

C. 1. LE CONTENU DE VOTRE APPLICATION :

- Lister les contenus que le prestataire doit reprendre :
  - L'api faites en local mais qui doit se baser sur leur système de gestion des données (celui du forum)
- Lister les contenus que le prestataire doit créer :
  - Publier l'application sur les stores
  - Acheter les licences pour avoir les droits de publication

C. 2. CONTRAINTES TECHNIQUES :

Précisez vos attentes concernant les besoins connexes à ce projet que le prestataire devra fournir :

- Hébergement de l'API C# .net 6.0
- Assurer la maintenance
- Proposer des astreintes pour le dépannage
- Formation à l'utilisation du back office

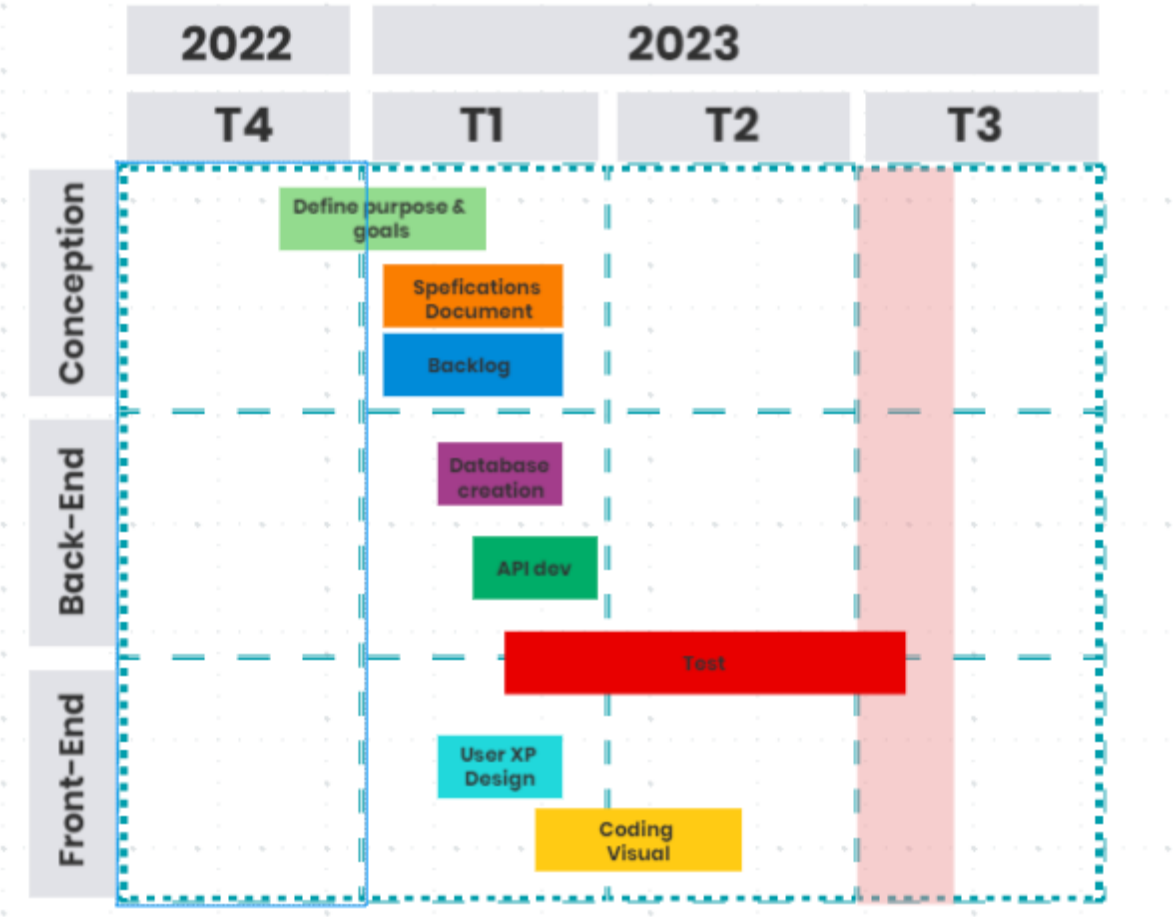
C. 3. LE PLANNING :

Ajoutez un agenda des dates souhaitées pour la validation des différentes étapes :

- Date de la création des maquettes : 9/1/2023
- Date de validation des maquettes : 10/1/2023
- Date de la création de l'API : 6/1/2023
- Date de validation de l'API : 10/2/2023
- Dates des tests : à partir d'avril 2023
- Date de mise en ligne sur les stores :

  à décider par les possesseur du site SMOGON

Gestion de projet :



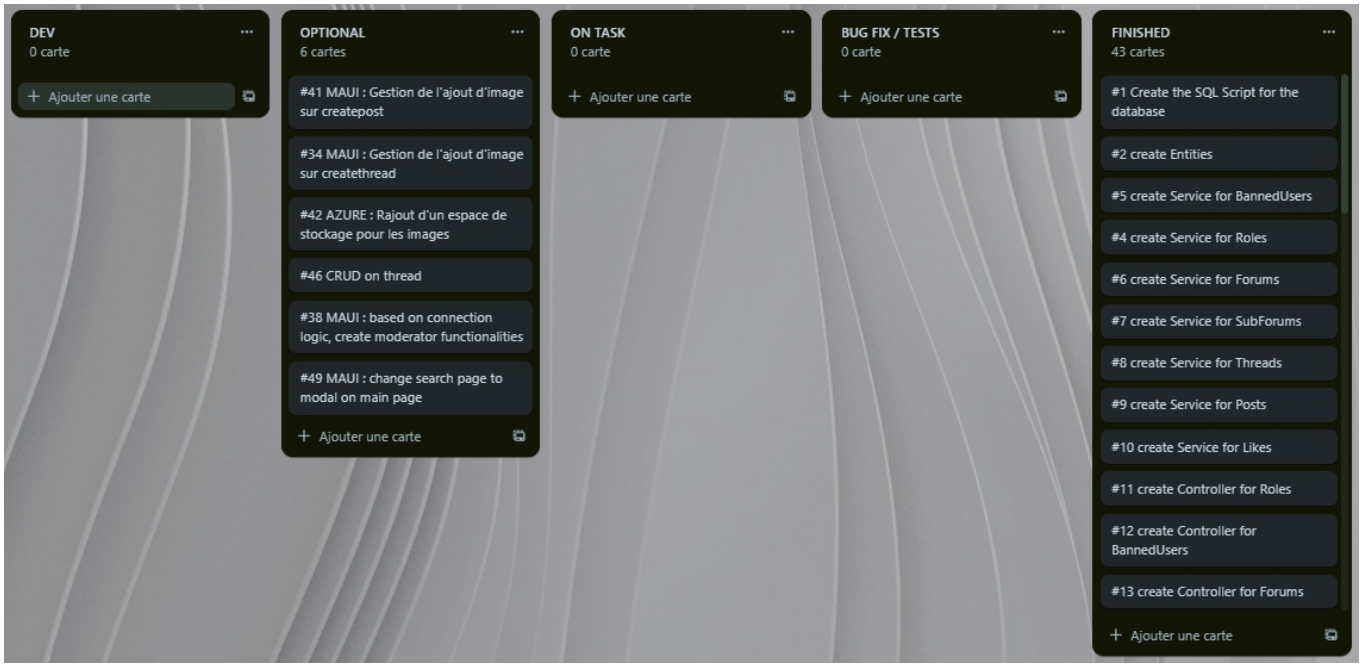
A l'origine du projet, nous avons mis en place le planning des différentes phases du développement ci-dessus. La majorité de la documentation que je vais vous exposer est en anglais, la communauté SMOGON étant internationale et ayant choisi ce langage comme vecteur de communication.

Nous avons conçu ensemble le cahier des charges qui combine les espace “DEFINE PURPOSE AND GOALS” et “SPECIFICATION DOCUMENT” que vous pouvez retrouver dans le cahier des charges.

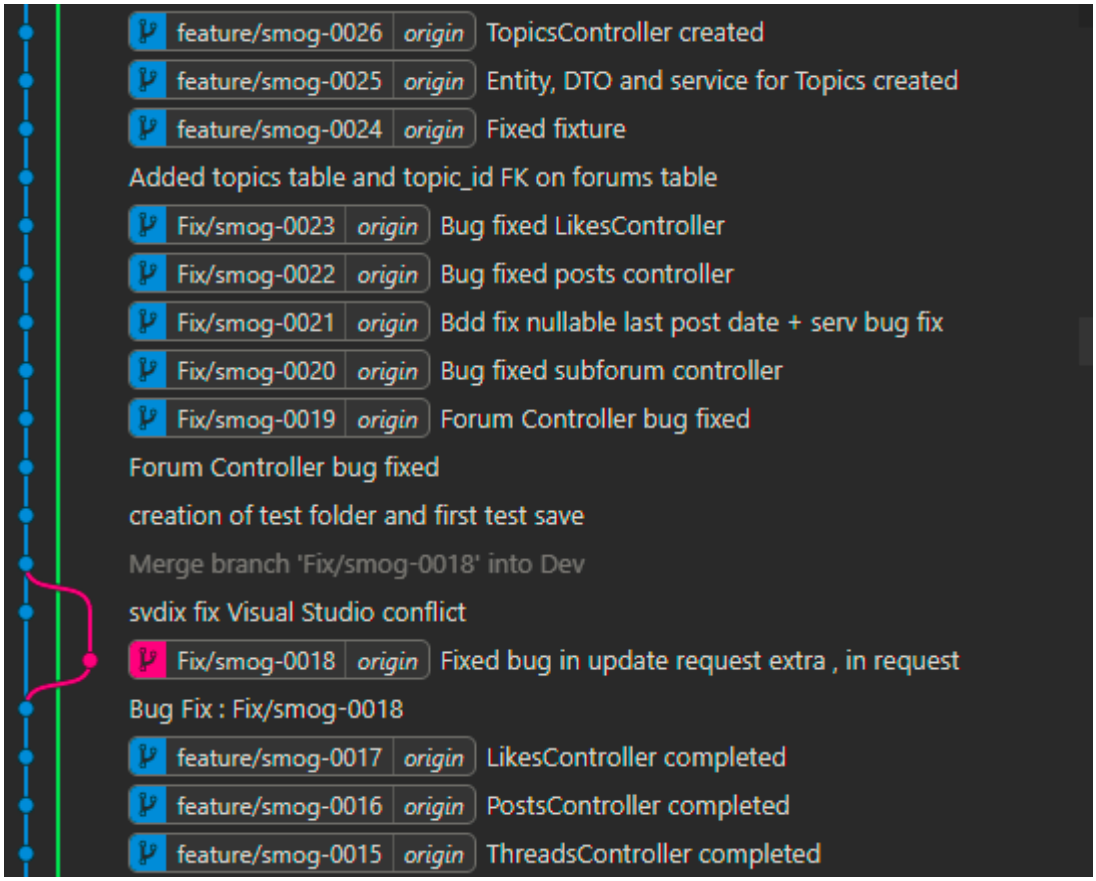
De plus, il a été décidé que dans l’optique d’une **maintenabilité** et d’une **lisibilité du projet**, on utiliserait le **minimum possible de librairies**, et, que l’on utiliserait une **norme pour l’écriture du code** que je vais détailler dans ce chapitre.

Pour que les “products owner” puissent avoir une vision du projet, il a été versionné sur Github et nous avons mis en place un espace de travail trello qui permettait la suivie du projet et des tickets.

Voici une capture d'écran de la page trello à la fin de la phase 1 de développement, qui a été fournie en amont, mais aussi parfois à volée, car certains tickets représentaient beaucoup de développement et ont été subdivisés :



Dans la capture d'écran suivante, nous voyons un représentation graphique de certaines entrée lors du versionning, on y retrouve les numéros de ticket pour les fonctionnalités listées sur trello.



En terme de choix de technologie, nous nous sommes orientés vers C#, .net, SQL et MAUI pour les raisons suivantes :

- **C#** est un langage de programmation puissant et expressif, permettant de développer un code clair et maintenable.
- **MAUI** (Multi-platform App UI) est un framework qui permet de créer des interfaces utilisateur multiplateformes attrayantes et réactives, avec un code partagé entre les différentes plateformes.
- **SQL** est un système de gestion de base de données robuste et largement utilisé, offrant une grande flexibilité pour la gestion des données de l'application.
- La **communauté de SMOGON** est une communauté qui compte beaucoup de développeur bénévoles et de néophytes habitués à bidouiller des applications desktop et des softs autour de l'univers du jeux-vidéo, qui sont très souvent en **C#**, le front est souvent en **WPF** et ils savent aussi souvent utiliser des **bases de données en SQL** (server ou mysql)

Nous avons ensuite établi les différentes fonctionnalités du site. Nous avons ainsi décidé de reproduire l'expérience du forum pour ne pas choquer le public habitué de longue date, mais de néanmoins inclure des fonctionnalités supplémentaires :

- **Comptes utilisateurs :** Les utilisateurs doivent pouvoir créer et gérer leurs propres comptes, y compris la configuration de leur profil et la mise à jour de leurs informations personnelles.
- **Fils de discussion et messages :** Le cœur de tout forum réside dans la possibilité de créer des fils de discussion (sujets) et de publier des messages (réponses) dans ces fils de discussion. Les utilisateurs doivent pouvoir créer de nouveaux fils de discussion et publier des messages, ainsi que modifier ou supprimer leurs propres messages.
- **Recherche :** Les utilisateurs doivent pouvoir rechercher des fils de discussion ou des messages spécifiques dans le forum.

- **Modération :** Des outils doivent être mis en place pour permettre aux modérateurs de gérer le contenu du forum, y compris la possibilité de supprimer ou modifier des messages, d'exclure des utilisateurs et d'établir des règles pour le forum.
- **Notifications :** Les utilisateurs doivent pouvoir recevoir des notifications lorsqu'une personne répond à l'un de leurs messages ou les mentionne dans un fil de discussion.
- **Profils d'utilisateurs :** Chaque utilisateur doit disposer de sa propre page de profil qui affiche son activité sur le forum, y compris les fils de discussion et les messages qu'il a créés.
- **Pièces jointes d'images et de fichiers :** Les utilisateurs doivent pouvoir joindre des images et des fichiers à leurs messages.
- **Création d'équipe :** Les utilisateurs doivent pouvoir publier la composition de leur équipe à partir d'un formulaire de texte.
- **Évaluation des messages :** Les utilisateurs doivent pouvoir voter positivement pour les messages.

Nous avons opté pour l'utilisation d'une API REST (Representational State Transfer), car elles offrent une architecture simple, basée sur les standards du web tels que HTTP et JSON. Elles permettent une communication efficace entre différentes parties de l'application, facilitent l'indépendance des déploiement, favorisent la flexibilité et l'évolutivité. Elles permettent de développer des applications modulaires et réutilisables, tout en suivant des bonnes pratiques de conception et de développement. En résumé, elles sont une approche fiable, largement adoptée et adaptée aux besoins actuels de développement d'applications.

Ainsi nous avons établi les routes suivantes pour cartographier l'intégralité des fonctionnalités :

- **Utilisateurs :**

```
GET /api/users              # Récupère la liste de tous les utilisateurs
GET /api/users/:id          # Récupère l'utilisateur par ID
POST /api/users             # Crée un nouvel utilisateur
PUT /api/users/:id          # Met à jour un utilisateur existant
DELETE /api/users/:id       # Supprime un utilisateur
```

POST /api/users/:id/login # Connecte un utilisateur
GET /api/users/:id/threads # Récupère tous les fils de discussion par ID utilisateur
GET /api/users/:id/posts # Récupère tous les messages créés par ID utilisateur

- Rôles :

GET /api/roles # Récupère la liste de tous les rôles
GET /api/roles/:id # Récupère le rôle par ID
POST /api/roles # Crée un nouveau rôle
PUT /api/roles/:id # Met à jour un rôle existant
DELETE /api/roles/:id # Supprime un rôle
GET /api/roles/:id/users # Récupère tous les utilisateurs par ID de rôle

- Forums :

GET /api/forums # Récupère la liste de tous les forums
GET /api/forums/:id # Récupère le forum par ID
POST /api/forums # Crée un nouveau forum
PUT /api/forums/:id # Met à jour un forum existant
DELETE /api/forums/:id # Supprime un forum
GET /api/forums/:id/subforums # Récupère tous les sous-forums par ID de forum
GET /api/forums/:id/threads # Récupère tous les fils de discussion par ID de forum

- Sous-Forum :

GET /api/subforums # Récupère la liste de tous les sous-forums
GET /api/subforums/:id # Récupère le sous-forum par ID
POST /api/subforums # Crée un nouveau sous-forum
PUT /api/subforums/:id # Met à jour un sous-forum existant
DELETE /api/subforums/:id # Supprime un sous-forum
GET /api/subforums/:id/threads # Récupère tous les fils de discussion par ID de sous-forum

- **Sujets:**

GET /api/threads # Récupère la liste de tous les fils de discussion
GET /api/threads/:id # Récupère le fil de discussion par ID
POST /api/threads # Crée un nouveau fil de discussion
PUT /api/threads/:id # Met à jour un fil de discussion existant
DELETE /api/threads/:id # Supprime un fil de discussion
GET /api/threads/:id/posts # Récupère la liste de tous les messages d'un fil de discussion

- **Posts :**

GET /api/posts # Récupère la liste de tous les messages
GET /api/posts/:id # Récupère le message par ID
POST /api/posts # Crée un nouveau message
PUT /api/posts/:id # Met à jour un message existant
DELETE /api/posts/:id # Supprime un message
GET /api/posts/:id/likes # Récupère la liste de tous les likes d'un message

- **Likes :**

GET /api/likes/:id # Récupère le like par ID
POST /api/posts/:id/likes # Permet à un utilisateur d'aimer un message
DELETE /api/posts/:id/likes/:like_id # Permet à un utilisateur de ne plus aimer un message

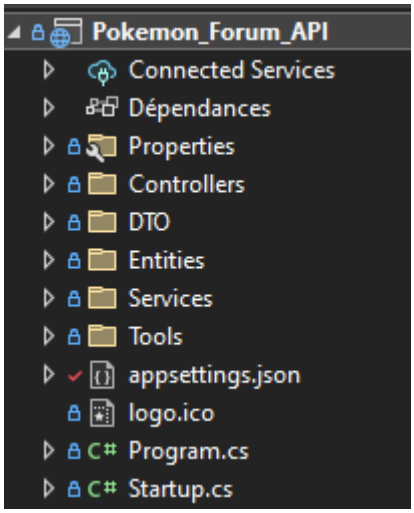
- **Utilisateurs Bannis:**

GET /bannedusers # Récupère tous les utilisateurs bannis
GET /bannedusers/:id # Récupère un utilisateur banni spécifique par ID
POST /bannedusers # Crée un nouvel utilisateur banni
PUT /bannedusers/:id # Met à jour un utilisateur banni existant
DELETE /bannedusers/:id # Supprime un utilisateur banni existant

Nous avons conçu les modèles de données en se basant sur ces besoins. Durant cette phase, J'ai mis en place les différents modèles, en m'appuyant et épurant le dictionnaire de données. Voici les différents modèles que j'ai réalisés dans l'ordre suivant (ces modèles sont disponibles en annexe) :

- Le modèle conceptuel de données (MCD) du type modèle entité-association. Qui permet de décrire le système d'information à l'aide d'entités.
- Modèle logique de données (MLD), qui consiste à décrire la structure selon laquelle les données seront stockées en dans la base de données

Nous avons choisi l'architecture suivante le projet en microservices car L'utilisation d'une architecture en microservices présente plusieurs avantages significatifs. Tout d'abord, elle favorise la modularité de l'application, en permettant de découper celle-ci en services indépendants, chacun se concentrant sur une fonctionnalité spécifique. Cette modularité facilite le développement et le déploiement de l'application, car les services peuvent être développés, testés et déployés de manière indépendante. De plus, cette approche permet une meilleure évolutivité, car chaque service peut être mis à l'échelle individuellement en fonction des besoins. L'architecture en microservices facilite également la réutilisation du code, car les services peuvent être partagés et utilisés par d'autres applications. De plus, en cas de défaillance d'un service, les autres services peuvent continuer à fonctionner, ce qui améliore la résilience globale de l'application. Enfin, l'architecture en microservices favorise une meilleure collaboration au sein des équipes de développement, car chaque service peut être développé par une équipe dédiée, ce qui permet une plus grande autonomie et une meilleure spécialisation. La notre ressemble à ceci :



Après plusieurs corrections d'erreur suite aux remontées de test, nous avons choisi de publier l'application et celà s’est fait, à notre grand regret, en deux phases.

Nous avons tout d’abord opté pour une publication sur azure de part la simplicité de l’interface de publication de Visual Studio, mais il s’est avéré que le coût était faramineux (+ de 200€ pour 1 semaine d’hébergement).

Puis nous avons choisi de prendre un serveur virtuel privé (VPS) avec la société IONOS. La publication de l’API s’est révélée tout autant facile grâce à l’interface Plesk. Cependant, au lieu de ne cliquer que sur un seul bouton, il nous faut compiler l’application puis la publier sur le site.

A ce jour, la publication de l'application sur les stores et en cours de discussion, les test foctionnels ont été fait et le rendu de la version 1 de l'application satisfait l'équipe.

Dans la phase de développement de la version 2, les pistes d'amélioration sont :

- l'implémentation de la fonction de recherche qui a été stoppée car la conception était plus que bancale
- l'ouverture de la discussion sur la possibilité de modération via l'application mobile qui n'a pas été mise en place malgré que les routes et les fonctionnalités soient elles-mêmes développées

RÉALISATION – EXTRAIT DE CODE :

Nous allons nous focaliser sur l'aspect connexion de l'utilisateur pour mettre en exergue plusieurs aspects fondamentaux du développement.

Le modèle de base pour un utilisateur est comme suit :

```
#region Properties

[Key]
[DatabaseGenerated(DatabaseGeneratedOption.Identity)]
7 références
public int user_id { get; set; }
13 références
public string username { get; set; }
7 références
public string password { get; set; }
6 références
public string email { get; set; }
4 références
public DateTime join_date { get; set; }
3 références
public string avatar_url {get; set; }
4 références
public bool isBanned { get; set; }
5 références
public int role_id { get; set; }
1 référence
public List<Posts> posts { get; set; }
1 référence
public List<Teams> teams { get; set; }

#endregion

#region Constructor

4 références
public Users(){}

6 références
public Users(int user_id, string username, string password, string email, DateTime join_date, string avatar_url, int role_id, bool isBanned )
{
    this.user_id = user_id;
    this.username = username;
    this.password = password;
    this.email = email;
    this.join_date = join_date;
    this.avatar_url = avatar_url;
    this.role_id = role_id;
    this.isBanned = isBanned;
}
```

Voici le controlleur relatif à la route User et nous allons nous focaliser sur la fonction de connexion :

```
[ApiController]
[Route("/users")]
1 référence
public class UsersController : ControllerBase
{
    string connectionString = Tools.Tools.connectionString;
    //string connectionString = Utils.ConnectionString;
    //string connectionString = @"Server=127.0.0.1;User ID=root;Password=;Database=smogon_forum;";
    UserService userService = new UserService();

    0 références
    public UsersController(){}

    [HttpGet]
    0 références
    public async Task<ActionResult<List<Users>>> GetAllUsers()
    {
        var users = await userService.GetAllUsers(connectionString);
        if (users == null)
            return BadRequest("An error occurred while getting all the users. Please check your request and try again.");
        return Ok(users);
    }
```

```
[HttpPost("login")]
0 références
public async Task<IActionResult> Login([FromBody] UserDtoLogin userDto)
{
    var token = await userService.LoginUserJWT(connectionString, userDto.username, userDto.password);
    if (token == null)
    {
        return Unauthorized();
    }

    return Ok(new { token = new JwtSecurityTokenHandler().WriteToken(token) });
}
```

On peut voir sur la capture d'écran ci-dessus que la méthode de login utilise le principe de token JWT, que je vais détailler un peu plus bas pour parler d'un aspect sécurité.

L'objet permettant l'accès aux données (nous avons choisi de différencier la classe de base du DTO pour une question de compréhension pour les néophytes) :

```
1 référence
public class UserDtoLogin
{
    [Required]
    [StringLength(20, MinimumLength = 3, ErrorMessage = "username length must be between 3 and 20")]
    1 référence
    public string username { get; set; }
    [Required]
    [StringLength(20, MinimumLength = 8, ErrorMessage = "password length must be between 8 and 20")]
    1 référence
    public string password { get; set; }

}
```

Une première couche de limitation est imposée par le DTO et une prise en charge d'erreur.

Ensuite, nous allons voir sur le service relatif à l'utilisateur ci-dessous:

```
19 références
public class UserService
{
    string connectionString = Tools.Tools.connectionString;
    9 références
    public UserService() {}
```

```
/// <summary>
/// Method to login User using JWT tokens
/// </summary>
/// <param name="connString"></param>
/// <param name="_username"></param>
/// <param name="_password"></param>
/// <returns></returns>
1 référence
public async Task<SecurityToken> LoginUserJWT(string connString, string _username, string _password)
{
    Users user = new Users();

    try
    {
        using (MySqlConnection conn = new MySqlConnection(connString))
        using (MySqlCommand cmd = new MySqlCommand("SELECT * FROM users where username=@username", conn))
        {
            await conn.OpenAsync();
            cmd.Parameters.AddWithValue("@username", _username);

            using (var reader = await cmd.ExecuteReaderAsync())
            {
                while (await reader.ReadAsync())
                {
                    int id = reader.GetInt32(0);
                    string username = reader.GetString(1);
                    string password = reader.GetString(2);
                    string email = reader.GetString(3);
                    DateTime join_date = reader.GetDateTime(5);
                    string avatar_url = reader.IsDBNull(4)? "www.exemple.fr/imagedefouf" : reader.GetString(5);
                    bool isBanned = reader.GetBoolean(6);
                    int role_id = reader.GetInt32(7);

                    if (BCrypt.Net.BCrypt.Verify(_password, password))
                    {
                        user = new Users(id, username, password, email, join_date, avatar_url, role_id, isBanned);
                    }
                    else
                    {
                        return null;
                    }
                }
            }
        }
```

La fonction admet trois paramètres: le chaînon de connexion, et deux chaînes de caractères relatifs au mot de passe et au nom d'utilisateur. Pour pallier aux injections SQL, les requêtes sont préparées puis exécutées avec les paramètres établis.

Parmis le peu de librairies utilisées nous avons choisi la libraire Bcrypt pour la prise en charge de la vérification des token JWT(JSON Web Tokens), qui sont utilisés pour l'authentification et l'autorisation dans les applications web et mobiles.

Les tokens JWT offrent une sécurité garantie par leur signature numérique et leur capacité à contenir toutes les informations nécessaires pour vérifier l'identité et les autorisations d'un utilisateur. Ils permettent une

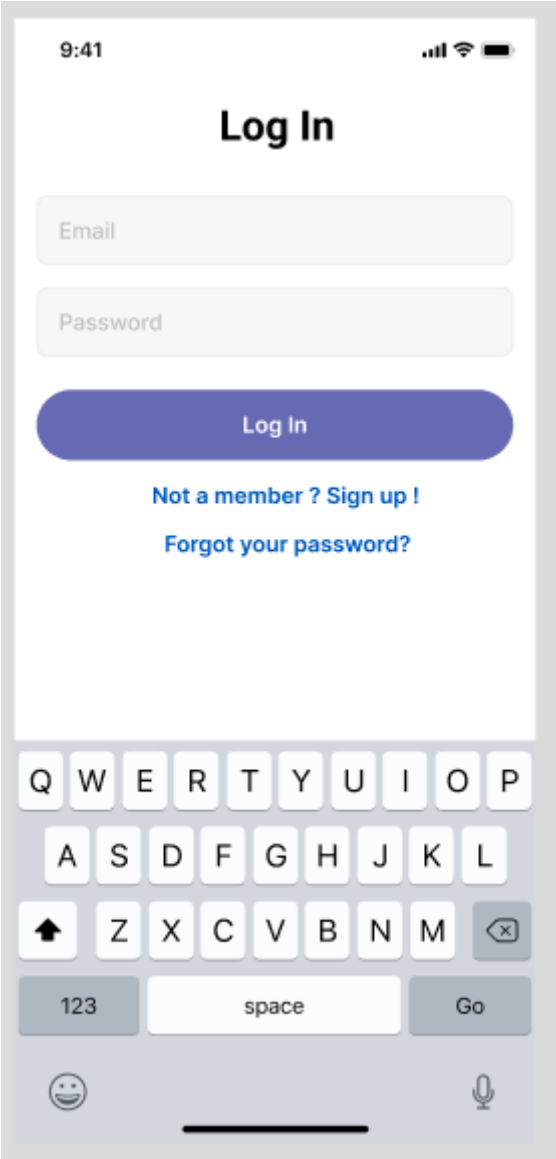
transmission efficace via divers moyens tels que les URL, les en-têtes HTTP ou les cookies. Les tokens JWT peuvent être configurés avec une durée de validité pour renforcer la sécurité. Grâce à leur auto-validation, ils améliorent les performances et facilitent le passage à l'échelle sans nécessiter des consultations constantes d'une base de données. Les tokens JWT sont compatibles avec diverses technologies et peuvent être personnalisés pour inclure des informations spécifiques à l'application.

Voici la deuxième partie de la fonction :

```
        if (user.user_id != 0)
        {
            //Getting the key from app setting
            IConfigurationBuilder builder = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("appSettings.json", optional: true, reloadOnChange: true);

            var configuration = builder.Build();

            JwtSettings jwtSettings = configuration.GetSection("Jwt").Get<JwtSettings>();

            var key = Encoding.ASCII.GetBytes(jwtSettings.Key);

            //Token shaping
            var tokenDescriptor = new SecurityTokenDescriptor
            {
                Subject = new ClaimsIdentity(new Claim[]
                    {
                        new Claim("User_id", user.user_id.ToString()),
                        new Claim("Role_id", user.role_id.ToString())
                    }),
                Expires = DateTime.UtcNow.AddDays(90),
                SigningCredentials =
                new SigningCredentials( new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature ),
                Issuer = jwtSettings.Issuer,
                Audience = jwtSettings.Audience
            };

            var tokenHandler = new JwtSecurityTokenHandler();
            var token = tokenHandler.CreateToken(tokenDescriptor);

            return token;
        }
        else
        {
            return null;
        }
    }
    catch (Exception ex)
    {
        return null;
    }
}
```

Cette partie sert à créer le token lors de l'authentification de l'utilisateur pour lui faciliter la connexion et l'interaction avec l'application. En cas de retour null, un message est alors envoyé par l'API pour dire que la connexion n'a pas pu être établie.

Cette fonctionnalité de l'api est utilisée sur la page d'accueil de l'application, dont je vais exposer le concept ci-dessous :



Dans le framework MAUI, les pages sont codées en utilisant une approche à deux parties distinctes. La première partie concerne l'aspect graphique en XAML, elle permet de définir la structure, la mise en page et l'apparence visuelle de la page. La deuxième partie concerne l'aspect logique, où les fonctionnalités et les comportements de la page sont implémentés. Cette séparation entre l'aspect graphique et l'aspect logique facilite la gestion et la maintenance du code, permettant aux développeurs de se concentrer sur des domaines spécifiques et de travailler de manière plus efficace.

Voici le bout de code de la partie XAML le plus significatif pour la fonction de login :

```
<Entry Grid.Row="1" HeightRequest="50"
       x:Name="usernameInput" Placeholder="Username"
       MaxLength="20"
       />

<Entry Grid.Row="2"
      HeightRequest="50"
      IsPassword="true"
       x:Name="passwordInput" Placeholder="Password"
       MaxLength="20"
       />

<Button Grid.Row="3" Text="Log in"
   Clicked="LoginUser"/>

<Button Grid.Row="4" Text="Forgot your password ?"
   BorderColor="Transparent" Background="Transparent"
   FontAttributes="Bold" TextColor="#094ABA"
   Clicked="PasswordForgottenAsync" />
```

Voici le bout de code de la partie XAML le plus significatif pour la fonction de login :

```
private async void LoginUser(object sender, EventArgs e)
{
    if(usernameInput.Text.IsNullOrEmpty() || passwordInput.Text.IsNullOrEmpty())
    {
        await DisplayAlert("Error", "All the fields must be filled in order to login", "OK");
    }
    else
    {
        var token = await userService.LoginUserJWT(usernameInput.Text.Trim(), passwordInput.Text.Trim());
        if(token != null)
        {
            Preferences.Set("token", token.RawData);
            Application.Current.MainPage = new AppShell();
        }
        else
        {
            await DisplayAlert("Error", "We have been unable to log you in with the information provided", "OK");
        }
    }
}
```

La fonction appelée vérifie le contenu des cellules, puis utilise un service pour tenter de connecter l'utilisateur et de récupérer le token JWT (généré par l'API comme vu plus haut). En cas d'erreur l'application affiche un message d'erreur à l'utilisateur.

Voici la partie de code importante du service user de l'application mobile :

```
public async Task<JwtSecurityToken> LoginUserJWT(string _username, string _password)
{
    try
    {
        // Create the request body
        var loginData = new Dictionary<string, string>
                                    {
                                        { "username", _username },
                                        { "password", _password }
                                    };

        var json = JsonConvert.SerializeObject(loginData);

        var content = new StringContent(json, Encoding.UTF8, "application/json");

        // Send the POST request
        var response = await client.PostAsync("users/login", content);

        if (response.IsSuccessStatusCode)
        {
            // Extract the token from the response
            var responseJson = await response.Content.ReadAsStringAsync();
            var loginResponse = JsonConvert.DeserializeObject<LoginResponse>(responseJson);
            var token = new JwtSecurityToken(loginResponse.Token);
            return token;
        }
        else
        {
            return null;
        }

    }
    catch (Exception ex)

    {
        return null;
    }
}
```

La fonction exploite les champs remplis en terme de mot de passe et de nom d'utilisateur, fabrique un objet json qui va servir de body. Ensuite, la fonction appelle la route que nous avons vu plus haut.

En cas de réponse avec un statut indiquant le bon déroulement de la requête, la fonction renvoie le token. La fonction est asynchrone et renvoie en fait une tache, pour éviter que l'application ne bloque dans l'attente d'une réponse.

```
if(token != null)
{
    Preferences.Set("token", token.RawData);
    Application.Current.MainPage = new AppShell();
}
```

Lorsque l'application récupère le token, elle le sauvegarde sur le téléphone de l'utilisateur et lance réellement l'application. Ce token est utilisé dans chaque phase de l'application lors des différents appels à l'API, c'est pourquoi le coeur de l'application n'est réellement démarré qu'une fois que le token a été reçu.

PHASE DE TEST :

Tests de l'API :

A l'issue du développement, nous avons écrit des scripts de test à utiliser sur l'application postman dont voici un extrait :


```
{
    "info": {
        "_postman_id": "049307ed-187d-4c9d-91b6-9171783bc96e",
        "name": "TESTS",
        "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
        "_exporter_id": "21702646"
    },
    "item": [
        {
            "name": "/users",
            "item": [
                {
                    "name": "Get all users",
                    "event": [
                        {
                            "listen": "test",
                            "script": {
                                "exec": [
                                    ""
                                ],
                                "type": "text/javascript"
                            }
                        }
                    ],
                    "protocolProfileBehavior": {
                        "disableBodyPruning": true
                    },
                    "request": {
                        "method": "GET",
                        "header": [],
                        "body": {
                            "mode": "raw",
                            "raw": "{\r\n    \"name\" : \"aurelie\",\r\n    \"age\" : 25\r\n}",
                            "options": {
                                "raw": {
                                    "language": "json"
                                }
                            }
```


Il suffisait alors à divers utilisateurs de renseigner l'adresse de l'api et de lancer les test puis de retourner les fichiers de log produit suite aux test.

Grâce à ces différents test l'application a pu évoluer et les bugs être corrigé en vu du déploiement.

Tests de l'application:

La compilation d'un executable pour les système androit est très aisée via l'environement de développement que nous avons choisi, et, un .apk a été distribué permettant aux utilisateurs de tester directement l'application.

L'API étant publiée au moment des test sur Azure, tout s'est passé pour le mieux. Il y a eu une remontée de test utilisateurs qui a été effectuée via des prises de notes, ou remarques faites directement. Ces tests ont permis la remontée de bug et problématiques en matière d'expérience utilsateur. L'application s'est bonnifée de cette expérience.

Néanmoins, n'ayant pas de compte apple devéloppeur, nous n'avons pas pu tester notre application sur IOS. Nous avons tenter d'utiliser un mac et de compiler le projet pour contourner la problématique, sans succès.

Ces tests nous ont aussi permis de réaliser la véritable puissante du framework .NET et de la facilité d'adaptation au divers environement du Framework MAUI.

VEILLE TECHNOLOGIQUE :

Lors de ma veille technologique portant sur le framework MAUI, j'ai examiné attentivement les aspects liés à la sécurité. La sécurité étant une préoccupation majeure dans le développement d'applications, il est essentiel de s'assurer que les frameworks utilisés sont robustes et protégés contre les vulnérabilités potentielles.

Dans le cadre de MAUI, j'ai étudié les versions les plus récentes et les correctifs de sécurité publiés. Il est encourageant de constater que Microsoft, le principal contributeur de MAUI, attache une grande importance à la sécurité et fournit régulièrement des mises à jour pour corriger les vulnérabilités identifiées.

Cependant, aucune technologie n'est à l'abri de vulnérabilités potentielles. J'ai identifié certaines vulnérabilités de sécurité signalées dans les versions précédentes de MAUI, telles que des failles d'injection de dépendances ou des problèmes de gestion de session. Cependant, il est important de noter que la plupart de ces vulnérabilités ont été corrigées dans les versions plus récentes du framework.

Pour garantir une sécurité optimale lors de l'utilisation de MAUI, il est recommandé de suivre les bonnes pratiques de sécurité, telles que la validation et l'échappement appropriés des données utilisateur, l'utilisation de mécanismes d'authentification et d'autorisation solides, ainsi que la protection contre les attaques courantes telles que les attaques par injection SQL ou XSS.

En outre, il est essentiel de maintenir MAUI et ses dépendances à jour en appliquant régulièrement les correctifs de sécurité publiés par les fournisseurs et en surveillant les avis de sécurité communautaires.

Dans l'ensemble, bien que MAUI bénéficie d'un engagement en matière de sécurité de la part de Microsoft et de la communauté des développeurs, il est crucial de rester vigilant, de suivre les meilleures pratiques et de maintenir à jour les bibliothèques et les dépendances pour réduire les risques de vulnérabilités de sécurité. Une veille continue et une collaboration avec la communauté des développeurs peuvent également contribuer à identifier et à résoudre rapidement les problèmes de sécurité émergents.

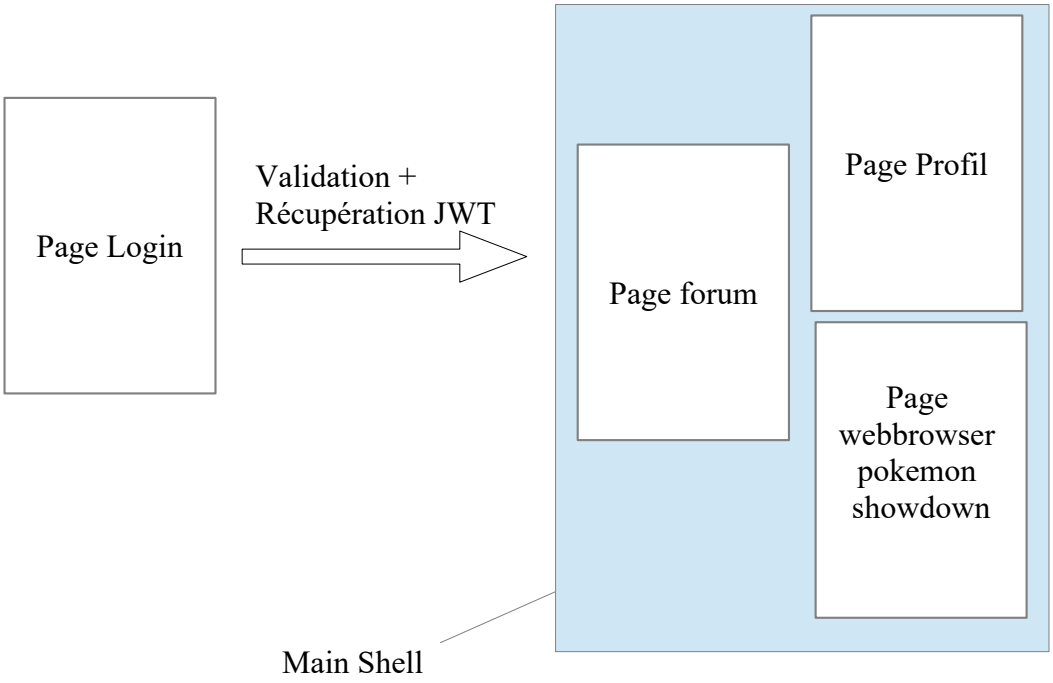
SITUATION DE TRAVAIL AYANT NÉCESSITÉ UNE RECHERCHE

Le framework MAUI étant jeune, la documentation ainsi que les recherches retournent peu souvent des résultats pertinant. Cependant, étant une évolution du framework Xamarin, certaines des réponses le concernant peuvent s'appliquer à l'environnement MAUI.

L'outil ChatGPT lui aussi est absolument démuni devant MAUI, et propose quasiment tout le temps des solutions totalement erronée.

De ce fait, j'ai pris le pas de poster des questions sur stackoverflow, communauté pertinante mais très peu souvent bienveillante, voir carrément élitiste.

Le cas s'est présenté lorsque j'ai essayé d'implémenter une navigation à l'intérieur d'une page, contenue elle même dans un élément de navigation. Pour plus de clarté je vais exposer la problématique sous-forme de schéma :



Mon but était de faire que la page profil elle-même devienne une sorte de sous-shell contenant plusieurs onglet à savoir les informations relatives à l'utilisateur, les équipes et les derniers posts, tout en cachant la top navigation bar qui apparaît automatiquement lorsqu'on créé un shell dans un shell. J'avais besoin dans ma conception, de placer la top navigation bar en dessous de la photo de profil.

J'ai donc ouvert la discussion suivante :

Hiding Top Navigations Bar in one of the shell tabs

Asked 4 months ago Modified 4 months ago Viewed 378 times

0

I've been trying to achieve hiding Top Navigations Bar in one of the shell tabs with no success.

I tried following this Tutorial with no success (might be outdated ?).

Here is my code :

```
                Route="Forum"/>
            <ShellContent
                Shell.NavBarIsVisible="False"
                NavigationPage.HasNavigationBar="False"
                ContentTemplate="{DataTemplate local:Pages.SubForum}"
                Route="SubForum"/>
            <ShellContent
                Shell.NavBarIsVisible="False"
                NavigationPage.HasNavigationBar="False"
                ContentTemplate="{DataTemplate local:Pages.Thread}"
                Route="Thread"/>
        </Tab>

        <Tab Icon="Resources/Images/search.png" Shell.NavBarIsVisible="False">
            <ShellContent
                ContentTemplate="{DataTemplate local:Pages.Search}"
                Route="Search">
            </ShellContent>
        </Tab>

        <Tab Icon="Resources/Images/snorlax.png" Shell.NavBarIsVisible="False">
```

J'ai ainsi obtenu cette réponse :

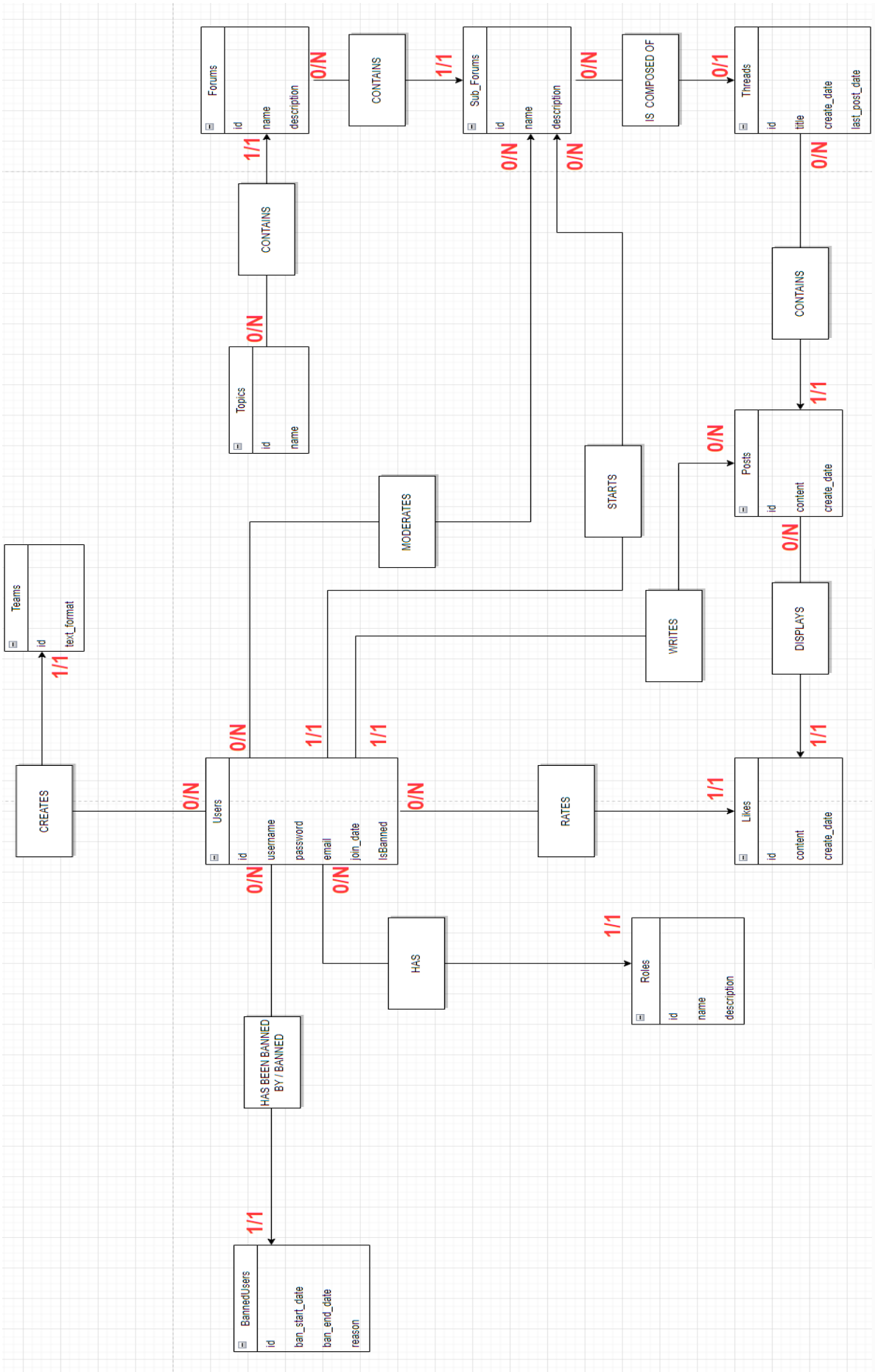
It is what i am saying it is, there is a bottom bar for shell navigation between the tabs, and this top navigation to navigate between the shellcontent of the first tab. You can't see the title for each sub tab because they are hidden, but you can click them anyway. – OzZzL Jan 24 at 14:56 Delete

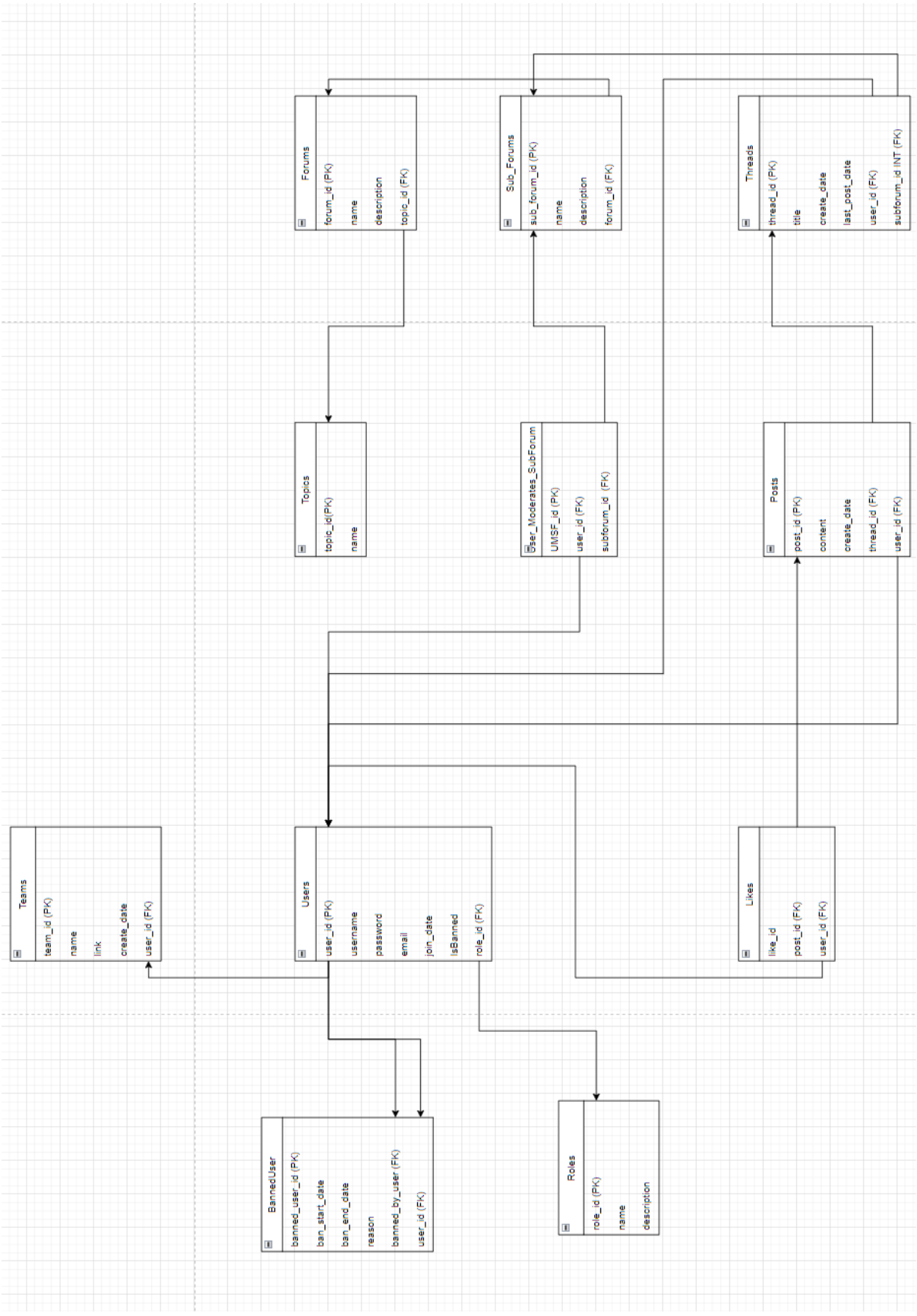
1 No. It is what I am saying it is. There is no such thing as "sub tab", and they are not "hidden, but you can click them anyway". It is a bar that has one purpose - to cycle between items in the list of ShellContents of that tab. You placed 4 ShellContents in that tab, the Shell is rendering this Bar to help you switch between them. You can write your own custom renderer, if you want to add Views, and then not render them. But each tab has its own navigation stack anyway. I see no reason for it. – H.A.H. Jan 24 at 15:19

@OzZzL Like H.A.H. said and what I've also mentioned in my comment below your question, you cannot do that (at least not easily). If you want bottom and top tabs, which is what you are using when adding multiple `<ShellContent>` items to a `<Tab>`, then you cannot get rid of the tab navigation, otherwise you wouldn't be able to navigate between those tabs: learn.microsoft.com/en-us/dotnet/maui/fundamentals/shell/... – Julian Jan 26 at 8:25

Il est ainsi apparu qu'il est actuellement très compliqué d'obtenir le résultat que je recherchais (je n'ai pas réussi), et j'ai donc dû revoir mes maquettes, et la conception.

ANNEXES :


Forums
id
name
description
CONTAINS
Sub_Forums
id
name
description
IS COMPOSED OF
Threads
id
title
create_date
last_post_date
CONTAINS
Topics
id
name
CONTAINS
Posts
id
content
create_date
CONTAINS
Teams
id
text_format
MODERATES
STARTS
WRITES
DISPLAYS
CREATES
Users
id
username
password
email
join_date
isBanned
RATES
Likes
id
content
create_date
HAS
Roles
id
name
description
HAS BEEN BANNED BY / BANNED
BannedUsers
id
ban_start_date
ban_end_date
reason
0/N
1/1


Forums
forum_id (PK)
name
description
topic_id (FK)
Sub_Forums
sub_forum_id (PK)
name
description
forum_id (FK)
Threads
thread_id (PK)
title
create_date
last_post_date
user_id (FK)
subforum_id INT (FK)
Topics
topic_id(PK)
name
User_Moderates_SubForum
UMSF_id (PK)
user_id (FK)
subforum_id (FK)
Posts
post_id (PK)
content
create_date
thread_id (FK)
user_id (FK)
Teams
team_id (PK)
name
link
create_date
user_id (FK)
Users
user_id (PK)
username
password
email
join_date
isBanned
role_id (FK)
Likes
like_id
post_id (FK)
user_id (FK)
BannedUser
banned_user_id (PK)
ban_start_date
ban_end_date
reason
banned_by_user (FK)
user_id (FK)
Roles
role_id (PK)
name
description