Je télécharge ma notice

Je télécharge la notice de mon appareil en que minutes via notre Annuaire

123Notices

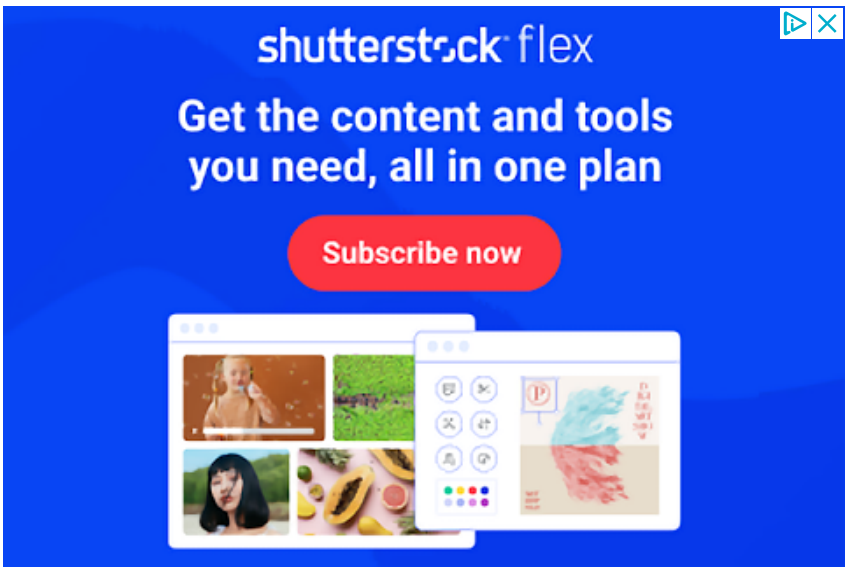
Téléc

Je télécharge ma notice

Je télécharge la notice de mon appareil en que minutes via notre Annuaire

123Notices

Téléc



Je télécharge ma notice

Je télécharge la notice de mon appareil en que
minutes via notre Annuaire

123Notices

Téléc

Je télécharge ma notice

Je télécharge la notice de mon appareil en que minutes via notre Annuaire

123Notices

Téléc

Je télécharge ma notice

Je télécharge la notice de mon appareil en que minutes via notre Annuaire

123Notices

Téléc

2. Précisez les moyens utilisés :

Les moyens utilisés pour ce projet :

- **Outil de modélisation**

J'ai utilisé **JMerise**, qui est un outil de modélisation de base de données. Il met à disposition la possibilité de créer des dictionnaires de données, de faire des MCD (Modèle Conceptuel de Données) et de vérifier si celui-ci est correcte. À partir, d'un MCD correct JMerise peut générer automatiquement le MLD (Modèle Logique de Données). C'est un outil de gain de temps.

- **SGBDR**

J'ai utilisé comme système de gestion de base de données relationnelles **PhpMyAdmin** qui incorpore le **langage SQL**.

- **Outil de gestion de projet**

Pour gérer ce projet, j'ai utilisé **GitLab**. Qui propose un hébergement gratuit, un système de versionning, et surtout un tableau de bord qui permet de planifier chacune des fonctionnalités de l'application sous forme d'étiquette (À faire, En cours, Fermer).

- **Documentations**

Je me suis documenté sur Symfony, voici deux sources qui me paraissent pertinentes :

- **https://symfony.com/doc/current/index.html#gsc.tab=0** : ce site met à disposition une documentation complète sur les bonnes pratiques et l'utilisation de Symfony, Doctrine et la création des classes qui permette le remplissage des tables (**Fixture**).
- Livre fourni par École EXPERNET qui s'intitule « **SYMFONY 2 Développez des sites web PHP structurés et performants** » : qui s'adresse aux développeurs, chef de projets, directeurs techniques ou, plus largement. Met à disposition l'apprentissage du Framework (Routage et contrôleur, les Templates avec Twig, Les bases de données avec Doctrine.

3. Avec qui avez-vous travaillé ?

J'ai travaillé avec :

- M. HOARAU Jérémie (Formateur), afin de définir ensemble les besoins. Il représentait le maitre d'ouvrage, et était force de propositions quant aux habitudes utilisateurs. Mais aussi pour la validation du Modèle Conceptuel de données.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ EXPERNET

Chantier, atelier, service ▸ Cadre de la formation : Développement Web côté serveur avec PHP et Symfony

Période d'exercice ▸ Du : 09/07/18 au : 20/07/18

5. Informations complémentaires *(facultatif)*

Activité-type 2 Développer la persistance des données

Exemple n° 2 ▸ Développement d'une application d'organisation de jeux– TINTOUR

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

TINTOUR est une application planification de jeux. Le concept est de permettre aux utilisateurs d'organiser des tournois et ainsi permettre une rencontre avec une date, une heure, un lieu et le jeu concerner (Billard, Baby-foot).

Dans un premier temps, après le recensement des besoins nous avons établi les différents cas d'utilisation. Ce qui nous a emmené à définir des règles gestion voici quelques-unes :

- **Un utilisateur (joueur) peut être propriétaire d'un à 5 salons.**
- **Un salon peut contenir un ou plusieurs utilisateurs (joueurs).**
- **Un utilisateur peut avoir un ou plusieurs ami(s) joueur.**
- **...**

Après plusieurs échanges avec les membres de l'équipe, j'ai recueilli diverses données, déterminé les données élémentaires, les données calculées, et les données composées en précisant la typologie de chaque donnée. Ce qui m'a permis de mettre en place rapidement un dictionnaire de données.

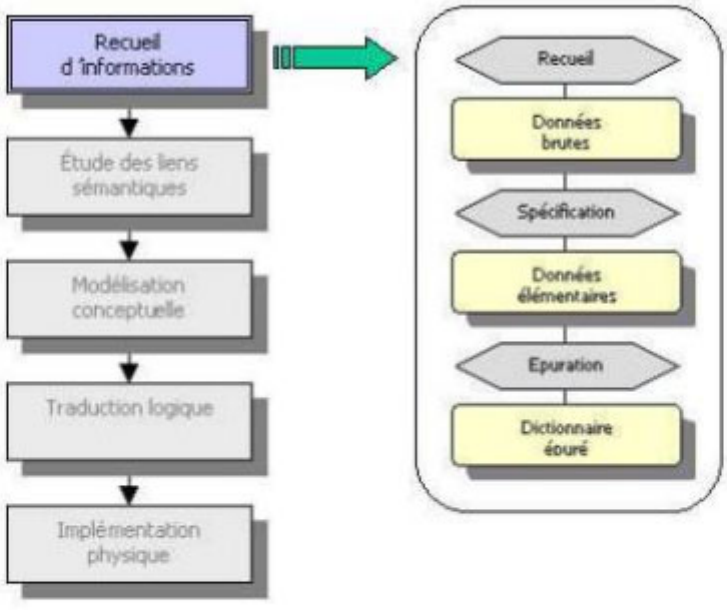
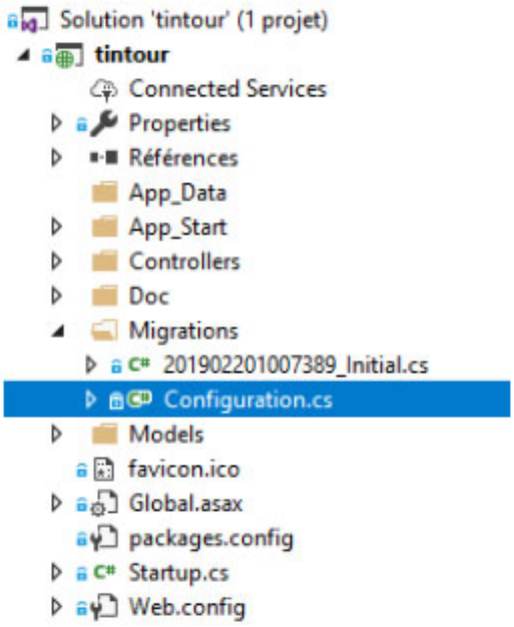


Figure 2 : Les phases de conception BDD

Dans un second temps, dans la conception technique, j'ai mis en place la phase de modélisation des données, en épurant ce dictionnaire mis en place précédemment.
Comme le montre la Figure 2, j'ai réalisé tout d'abord la modélisation conceptuelle, ensuite la modélisation logique et par la suite la modélisation physique des données.
Pour finir cette conception technique, j'ai représenté l'abstrait des objets du système qui vont interagir pour réaliser le cas d'utilisation par le biais d'un diagramme de classes.

Durant la phase de réalisation, j'ai créé les différentes classes de type entité. Puis, j'ai généré la base de données opérationnelle avec Entity Framework sous ASP.NET de Microsoft Visual Studio. Les tables sont créées sous SQL Server[2]. C'est un mécanisme qui se nomme Code First.
L'activation de la migration pour la création de la base de données, créée un nouveau dossier Migrations dans l'application contenant un fichier Configuration.cs. Permettant de définir le comportement de la migration pour le DbContext utilisé.



Dans la fonction ***Seed()*** de la classe Configuration, j'ai mis en place le script de jeux d'essai de la base de données. Au lancement d'une nouvelle commande de migration, les tables se remplissent automatiquement grâce à ce scripte.

[2] Microsoft SQL Server est un système de gestion de base de données en langage SQL incorporant entre autres un SGBDR développé et commercialisé par la société Microsoft.

```
protected override void Seed(tintour.Models.DbCtxt context)
{
    //  This method will be called after migrating to the latest version.

    //  You can use the DbSet<T>.AddOrUpdate() helper extension method
    //  to avoid creating duplicate seed data.

    //Players
    context.Player.AddOrUpdate(x => x.Id,
        new Player() { LastName = "Austen", FirstName = "Jane", Email = "jane@gmail.com" , Password = "jane", InRoom
        new Player() { LastName = "Tiphanie", FirstName = "Tremayne", Email = "ttremaynei@blogger.com", Password = "
        new Player() { LastName = "Gil", FirstName = "Hirth", Email = "ghirth2@themeforest.net", Password = "fvirnnn
        new Player() { LastName = "Ofilia", FirstName = "Tythe", Email = "otythe3@umich.edu", Password = "oTsj4S", I
        new Player() { LastName = "Zsazsa", FirstName = "Stockle", Email = "zstockle4@nationalgeographic.com", Passw
        new Player() { LastName = "Isiahi", FirstName = "Antuk", Email = "iantuk5@printfriendly.com", Password = "ce
        new Player() { LastName = "Caldwell", FirstName = "Faltin", Email = "cfaltin6@sakura.ne.jp", Password = "WKk
        new Player() { LastName = "Carry", FirstName = "Dowman", Email = "cdowman7@tinypic.com", Password = "HRhBSP"
        new Player() { LastName = "Dorothea", FirstName = "Thews", Email = "dthews8@squarespace.com", Password = "l8
        new Player() { LastName = "Tybi", FirstName = "Seiler", Email = "tseiler9@google.es", Password = "QKekuNFBaw
    );

    //Rooms
    context.Room.AddOrUpdate(r => r.Id,
       new Room() { CreatedAt = new DateTime(2019,01,01), Label = "Team974", MaxPlayers = 6, Location = "St Pierre",
       new Room() { CreatedAt = new DateTime(2019,01,15), Label = "Les Aigles", MaxPlayers = 4, Location = "St Leu",
       new Room() { CreatedAt = new DateTime(2019,01,05), Label = "LéBoss", MaxPlayers = 8, Location = "Le Port", Is
       new Room() { CreatedAt = new DateTime(2019,02,10), Label = "LéTab", MaxPlayers = 2, Location = "St Joseph", I
    );

    //Game
    context.Game.AddOrUpdate(g => g.Id,
        new Game() { Label = "Billard" },
        new Game() { Label = "Baby-Foot" }
    );

    //Friends
    context.FriendsInPlayers.AddOrUpdate(f => f.Id,
        new FriendsInPlayers() { PlayerId = 1, FriendId = 2 },
        new FriendsInPlayers() { PlayerId = 2, FriendId = 3 },
        new FriendsInPlayers() { PlayerId = 3, FriendId = 4)
        );
```

Côté client, je réalise deux boutons vue administrateur « Sauvegarde » et « Restauration ». Ensuite deux services.ts avec un appel direct à l'API.

```
Private _url = 'http://localhost:8080/api/BackupConfiguration';
Private _url = 'http://localhost:8080/api/RestoreConfiguration';
```

Côté serveur, je crée un nouveau contrôleur BackUpController.cs, j'implémente une nouvelle route.

```
[Route("api/BackupConfiguration")]
[Route("api/RestoreConfiguration")]
```

À partir de la règle de gestion suivante « **Un utilisateur (joueur) peut être propriétaire d'un à 5 salons** », avec l'équipe on a décidé de mettre en place un trigger qui se déclenche lors de la création d'un salon par un utilisateur, qui va compter le nombre de salon dans la table Rooms dont l'identifiant du joueur propriétaire est égal à celui de l'utilisateur qui réalise l'action de création du salon. Si, le résultat obtenu est inférieur à 5, le salon est créé. Sinon, un message d'erreur est retourné « nombre de salon trop important ».
Comme procédure stocker, nous avons opté pour une simple mise en majuscules des intitulés après création d'un salon.

2. Précisez les moyens utilisés :

Les moyens utilisés pour ce projet :

- **Editeur de texte**

J'ai utilisé **Visual Studio Code**, côté *Client* de l'application qui propose des extensions **Angular**, de bonne pratique. Et permet de se connecter au l'hébergeur **GitLab** ou **GitHub**.
Côté *Serveur*, j'ai utilisé **Microsoft Visual Studio**. Célèbre IDE, qui propose une suite de logiciels de développement. Dont dans ce projet, on a utilisé **ASP.NET** qui est un Framework de gestion à la demande des pages web.

- **Chrome**

Évidemment, pour visualiser le résultat du développement et valider des tests utilisateurs j'utilise comme navigateur **Chrome**.

- **SGBDR**

J'ai utilisé comme système de gestion de base de données relationnelles **Microsoft SQL Server** qui incorpore le **langage SQL**.

- **Outils collaboratifs et de gestion**

Ce projet a été réaliser en équipe, pour faciliter nos échanges.
Nous avons utilisé :

- **GitLab** : qui a permis d'avoir une base de projet identique. Avec le système de branche, chacun a pu travail sur différentes fonctionnalités. De plus, le système de CI/CD (système d'intégration continue) réduit les marges d'erreur de l'application. GitLab propose un tableau de bord, ce qui nous permet de faire la gestion du projet, planifier les différentes tâches à réaliser et les répartir.
- **Google Drive** : nous avons permis d'échanger les différents documents techniques, de travail collaborativement sur chaque présentation.

- **Documentations**

ASP.NET
http://www.eni-training.com/cs/expernet/olr

Entity framework
https://docs.microsoft.com/en-us/ef/core/

Angular 7.3.5
https://angular.io/docs

Angular material
https://material.angular.io/components/categories

3. Avec qui avez-vous travaillé ?

J'ai travaillé en équipe avec des étudiants de la promotion CDI (Concepteur Développeur Informatique) travaillant dans différentes entreprises suivantes :

- Hoarau John : Groupe MDSI
- Bastide Sébastien : TEEO
- Coli-Jun-Ong Kenny : De la mutualité de la Réunion

Nous constituons les maîtres d'œuvre, avec Hoareau John comme chef de projet et Bastide Sébastien, Coli-Jun-Ong Kenny, moi-même en tant qu'équipiers de projet.

L'application s'adresse aux étudiants du centre de formation d'Expernet, qui sont donc les maîtres d'ouvrage et nos utilisateurs Béta.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ EXPERNET

Chantier, atelier, service ▸ Dans le cadre de la formation : Développement d'une application mobile ECF 3

Période d'exercice ▸ Du : 23/04/19 au : 26/04/19

5. Informations complémentaires *(facultatif)*

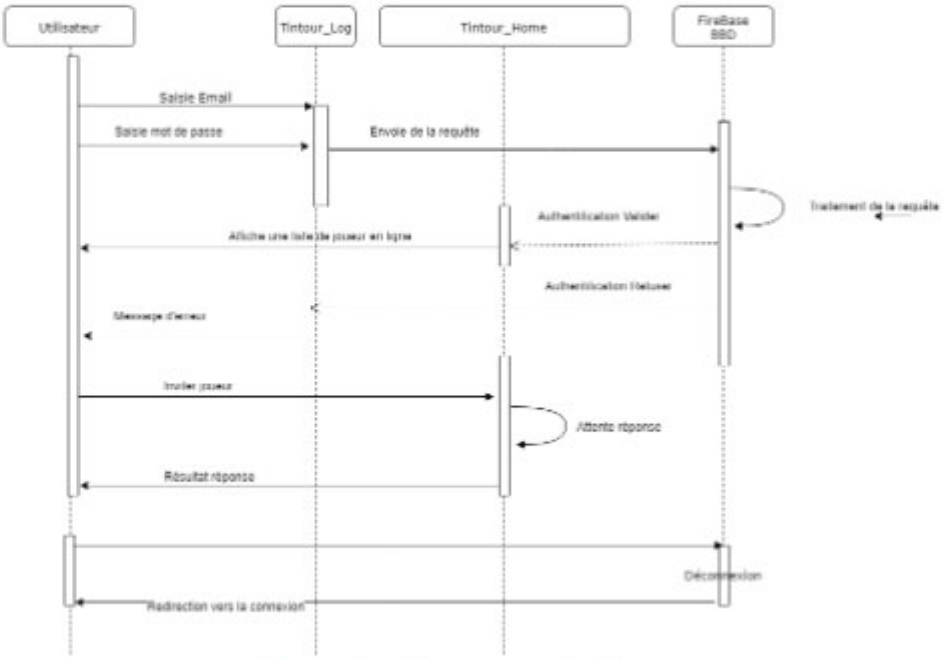


Figure 4 - Diagramme de séquence

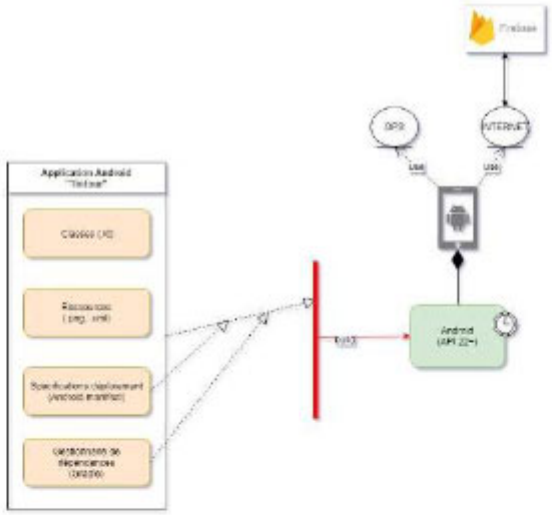


Figure 5- Diagramme de déploiement

Réalisation :

- Composants métier :

Nous avons développé, tout d'abord notre classe Player (Figure 5). Ensuite, nous avons mis en place le système de communication a Firebase (Figure 6). Étant donné que nous utilisons le système de géolocalisation de l'appareil mobile, il était indispensable de développer une classe permettant de récupérer cette distance et de déterminer la proximité des joueurs.

```kotlin
package com.expernet.tintour.models


/**
 * @class Player
 * Represents a player
 */
data class Player(
    val id          : String,
    val email       : String,
    var password    : String,
    val lastName    : String,
    val firstName   : String,
    var isOnline    : Boolean
) {
```

Figure 6 - Classe player

```kotlin
package com.expernet.tintour.models

import ...


class LocationListener : LocationListener {
    override fun onLocationChanged(location: Location?) {
        val long : Double? = location?.longitude
        val lat : Double? = location?.latitude

        if(long != null && lat != null) {
            updateLocation(long, lat)
        } else {
            updateLocation(MainActivityInterface.lastPosition.longitude, MainActivityInterface.lastPosition.latitude)
        }

        Log.d( tag: "LOC_CHANGED_1", OnlineUser.onlineUsers.toString()) // DEBUG
    }
```

Figure 7 - Classe de Géolocalisation

```kotlin
object OnlineUserManager{

    // Write a message to the database
    val database = FirebaseDatabase.getInstance()
    val myRef = database.getReference(OnlineUser::class.java.simpleName)

    /**
     * Saves OnlineUsers.onlineUsers to firebase
     */
    fun save() {
        myRef.setValue(OnlineUser.onlineUsers)
    }

    fun add(user: OnlineUser) {
//        myRef.child(user.id).setValue(user)
        myRef.child(user.id).setValue(user)
    }

    fun delete(uid: String) {
        myRef.child(uid).removeValue()
    }
```

Figure 8 -Connexion et interaction avec la base de données

Outil collaboratif et de version :

Durant ce projet, étant donné qu'ont travaillé en groupe, il était indispensable de se répartir les tâches pour le bon avancement du projet. Dans cette optique, nous savons mit en place un repositorie sur GitLab du projet, GitLab est un outil de gestion de projet qui permet de gérer ces