[**1. Introduction**](#_h77d0uz7qw9f) **4**

[**2. Contenu**](#_h77d0uz7qw9f) **4**

[2.1 Sujet](#_s1pk2ic2yo5h) 4

[2.2 But](#_3yzo3ia4dpmu) 4

[2.3 Spécifications](#_5o0alrx6udbx) 4

[2.4 Environnement de développement](#_kgejjquuz93v) 4

[2.5 Livrables](#_5o0alrx6udbx) 4

[**3. Analyse fonctionnelle**](#_h77d0uz7qw9f) **4**

[3.0 Profils utilisateur](#_415cnrw4hsfo) 4

[3.0Bis Les plans de navigation](#_1b7xncd2ipeh) 5

[3.0Bis.1 Visiteur(site web)](#_ojsm98f42kb1) 5

[3.0Bis.2 Utilisateur(site web)](#_5vyv791bod4j) 5

[3.0Bis.3 Visiteur(Application Android)](#_sg7wxyhkujoc) 6

[3.1 Partie visiteur(site web)](#_p1vdypews59) 7

[3.1.1 Connexion](#_ky2ilpfglydi) 7

[3.1.2 Enregistrement](#_hjt22g7pb0xh) 8

[3.2 Partie utilisateur(site web)](#_513fnjs4rnl2) 9

[3.2.1 Nav](#_byh3hgszwzao) 9

[3.2.2 Tableau de bord](#_5m22aooz45n5) 9

[3.2.3 Création d’événement](#_oepwza1q8auq) 10

[3.2.4 Modification d’événement](#_s9h1807gv7ih) 11

[3.2.5 Gérer l’événement(Pas encore démarré)](#_i23wtmgerkgj) 12

[3.2.6 Gérer l’événement(En cours)](#_oyedvfm8p66z) 13

[3.2.7 Gérer l’événement(Passé)](#_gxbrovdgp9ba) 14

[3.3 Partie visiteur(Application Android)](#_atc7hhymnlfj) 15

[3.3.1 Afficher événements En cours/Pas encore démarrés](#_d2uiv0dvnb97) 15

[3.3.2 Afficher événements Passés](#_wctz10r373d8) 16

[3.3.3 Détails d’événement(Pas encore démarré)](#_1ar3qobik3bx) 17

[3.3.3 Détails d’événement(En cours)](#_fn6zutspktvo) 18

[3.3.3 Détails d’événement(Passé)](#_6vta6ev34s17) 19

[**4. Analyse organique**](#_h77d0uz7qw9f) **20**

[4.1 Modèle physique (Modèle relationnel)](#_pi2ltqk45q3m) 20

[4.1.1 Users](#_wsu9177yrhfe) 20

[4.1.2 Events](#_m29dudmccf5k) 21

[4.1.3 Countries](#_24fjqwza3qzt) 21

[4.1.4 Messages](#_mlbx5anetvfk) 22

[4.1.4 Event\_States](#_a2ebp0h6pb0g) 22

[4.2 Classes utilisées](#_3c21pddd3jp4) 22

[4.2.1 Event](#_z6dwv4wodfo2) 22

[4.2.2 Message](#_p52ofxi8291s) 24

[4.2.3 EventState](#_9vbvmfbbs17g) 24

[4.2.4 Country](#_m46hrd2qh2le) 24

[4.3 Manager utilisés](#_dg7o1yle6oj4) 24

[4.3.1 UserManager](#_cb16z9ry6yqh) 24

[connect($nickname,$passwd)](#_6wa4fjliunim) 24

[userExist($nickname)](#_cp5yncih9il) 25

[createUser($nickname,$email,$passwd)](#_bf7yf3f30uht) 25

[validateUser($nickname,$token)](#_22ai07zah25p) 25

[4.3.2 SessionManager](#_lrupcj263rcf) 25

[addNickname($nickname)](#_lwnrf8wpafu4) 25

[getNickname()](#_upv87p2klk03) 25

[destroySession()](#_7s2zdm99a8k3) 25

[4.3.3 EmailManager](#_yh7tr6n6sbq) 26

[sendEmail($to,$subject,$text)](#_f1g3j0ngssbt) 26

[4.3.4 EventManager](#_ydyz4gts771u) 26

[getAllVisibleEvents($filter)](#_j6rwqoyqobky) 26

[getUserEvents($nickname,$filter)](#_obd0vt4zx26a) 26

[getMessages($eventId)](#_px2lljwlfyxr) 26

[addMessage($message,$eventId)](#_jovyc44u8gxs) 27

[createEvent($nickname,$title,$description,$country,$startDateTime)](#_7z75cpsy401a) 27

[updateEvent($eventId,$title,$description,$country,$startDateTime,$isVisible)](#_y8tf9tqv6yso) 27

[4.3.5 CountryManager](#_y35k8r5dhpt9) 28

[getLabel($iso)](#_o8c7msu0z2fd) 28

[getCode($label)](#_rbj19qh5l4u) 28

[getAllCountriesLabels()](#_vs4a7ux9hdcx) 28

[4.3.6 EventStateManager](#_yj5yij0kpcj) 28

[getLabel($code)](#_v4kss3dmsjxd) 28

[4.4 Class Database](#_gyte4vd5lxk6) 28

[4.4.1 Variable](#_idy8lrtbfk1e) 28

[4.4.2 Méthodes](#_dgrdbw1u4as) 29

[**5. Tests**](#_h77d0uz7qw9f) **29**

[**6. Planning prévisionnel**](#_h77d0uz7qw9f) **29**

[**7. Planning effectif**](#_h77d0uz7qw9f) **29**

[**8. Conclusion**](#_h77d0uz7qw9f) **29**

[**9. Glossaire**](#_h77d0uz7qw9f) **29**

# 1. Introduction

Pour le Travail Pratique Individuel (TPI), nous avons connu des changements concernant son déroulement, ce TPI a en effet été réalisé à 100% en télétravail suite aux mesures prise suite à l’épidémie de COVID-19.

Mon chef de projet pour ce travail est M. Aigroz. La personne m’ayant fourni le travail à réaliser est M. Court, responsable TPI.

# 2. Contenu

## 2.1 Sujet

Création d’un site web de création/suppression/gestion d’événements ainsi qu’une application mobile Android de consultation d’événement en live.  
Le projet se nomme Live events(Trad: Événements en direct).

## 2.2 But

Le but étant que des utilisateurs enregistrés puisse créer/supprimer/modifier des événements et gérer ces derniers en les alimentant de messages(flux) visibles live par des utilisateurs non-authentifiés depuis une application android.

## 2.3 Spécifications

## 2.4 Environnement de développement

* Un PC standard école, 2 écrans
* Windows 10
* Apache + module php,MySQL Server
* MySQL Workbench
* Visual Studio Code + extensions
* Visual Studio 2019 community edition + Xamarin
* Suite Office
* 1 émulateur Android

## 2.5 Livrables

* Planning
* Documentation technique
* Le résumé du TPI
* Manuel utilisateur
* Journal de bord

# 

# 3. Analyse fonctionnelle

## 3.0 Profils utilisateur

Le site web gère 2 profils de personne : visiteur et utilisateur.

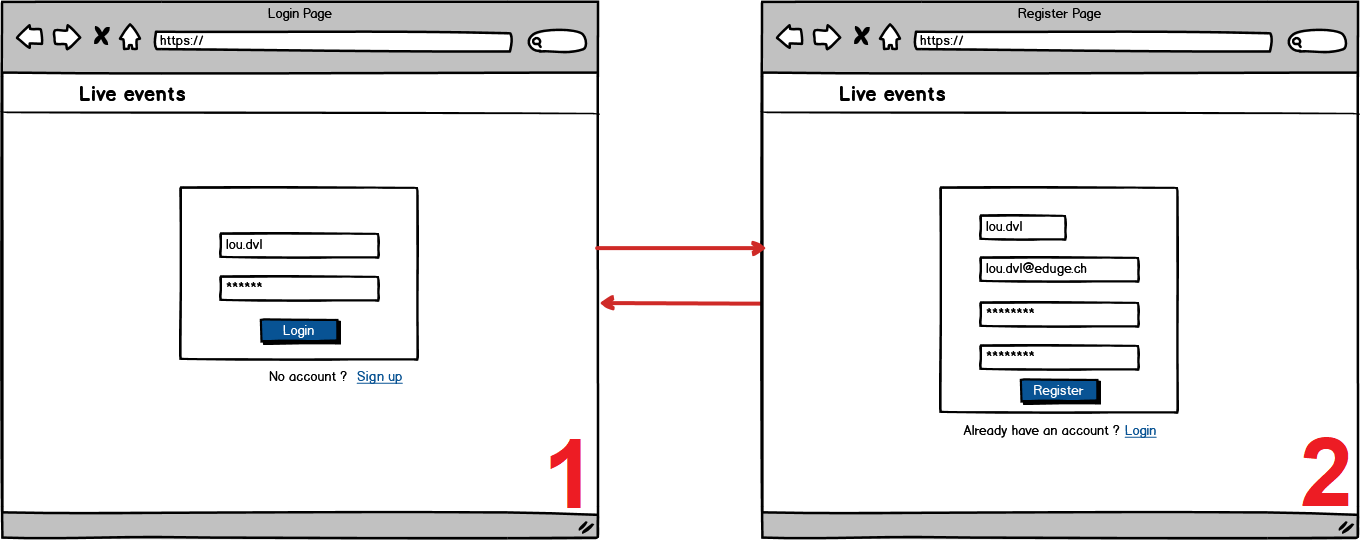
L’application mobile ne gère qu’un type de profil : visiteur

Chacun de ces profils donne accès à des fonctionnalités qui seront décrites ci-dessous.

## 3.0Bis Les plans de navigation

### 3.0Bis.1 Visiteur(site web)

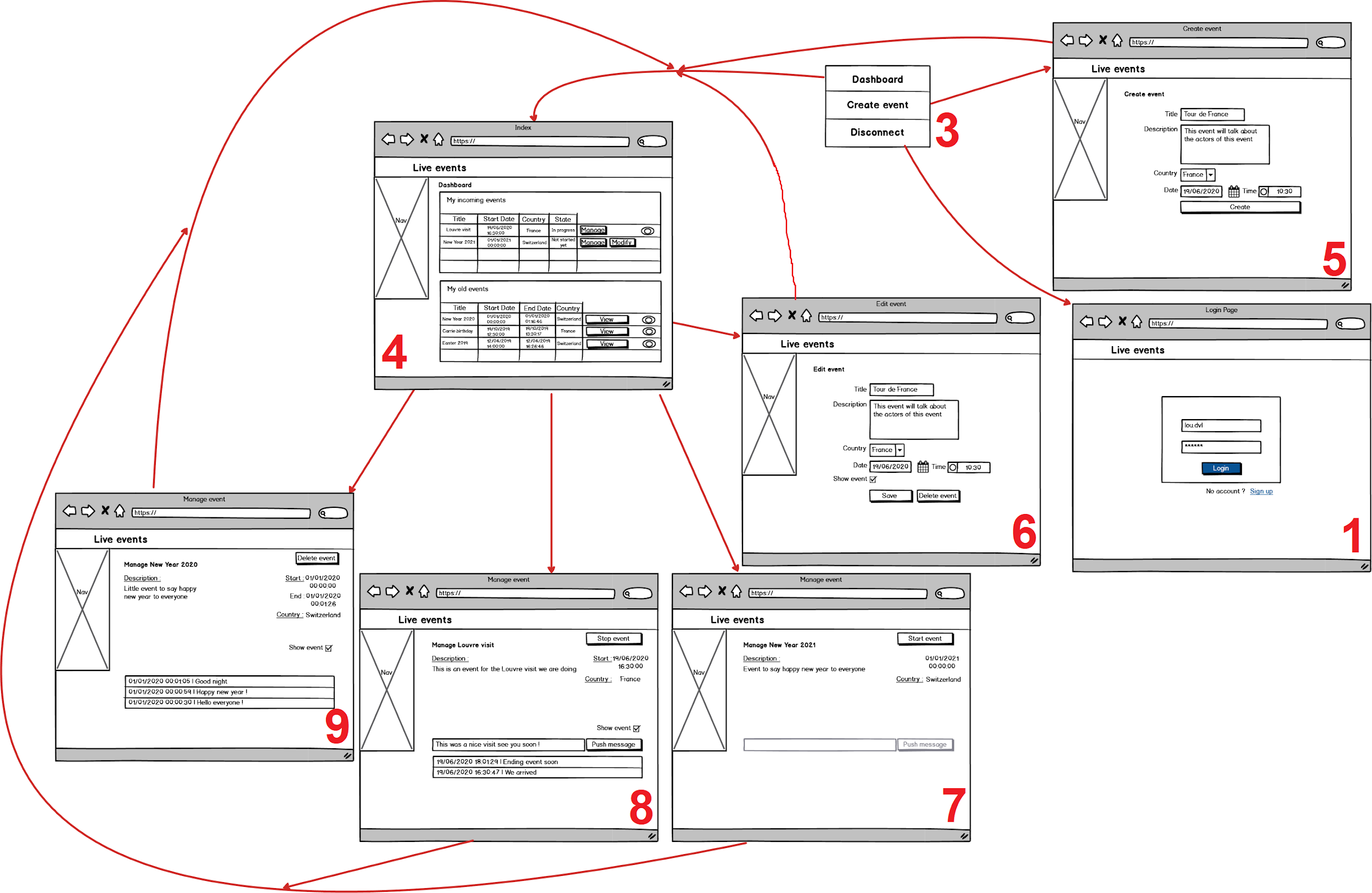
Le visiteur ne peut que se connecter ou se créer un compte à l’aide des pages login et register.



### 

### 3.0Bis.2 Utilisateur(site web)

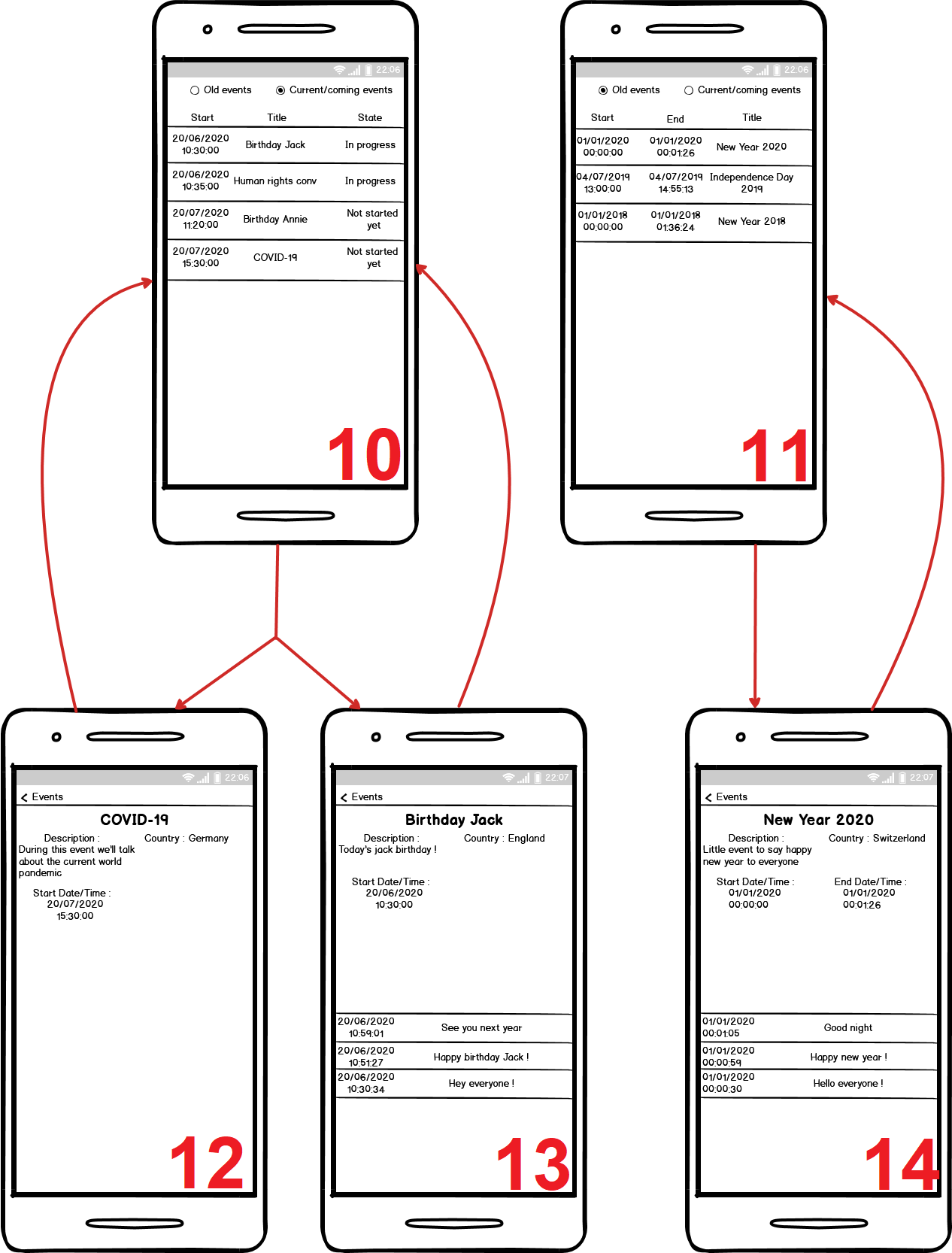
L’utilisateur a accès au tableau de bord, à la création, modification, gestion d’un événement.  
Lorsque l’utilisateur se déconnecte il est redirigé vers la page de connexion et repasse en profil visiteur.



### 

### 3.0Bis.3 Visiteur(Application Android)

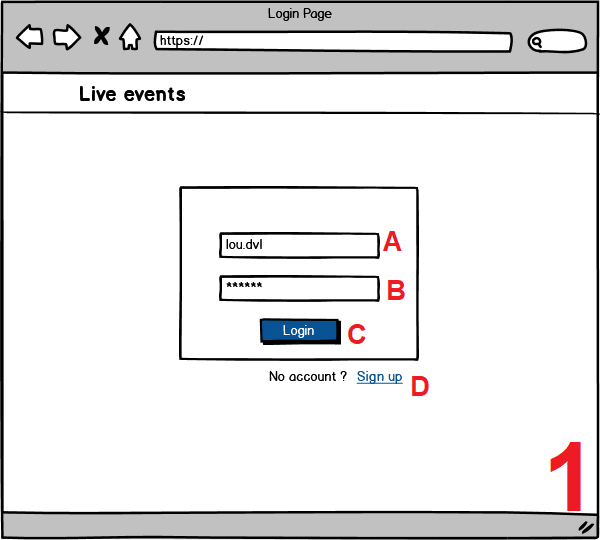
Le visiteur arrive sur la page de consultation de ses événements(figure 10) il peut changer d’affichage entre a figure 10 et 11 avec les radiobutton, il arrive sur la figure 12/13 ou 14 en fonction de l’événement sur lequel il clic, depuis ces écrans il peut retourner sur la figure 10/11 en cliquant sur le bouton “Events”.



## 3.1 Partie visiteur(site web)

### 3.1.1 Connexion

La page permettant à un utilisateur de se connecter avec ses identifiants, s’il n’a pas de compte il peut toujours cliquer sur le lien “Sign up” afin d’être rediriger vers la page de création de compte.

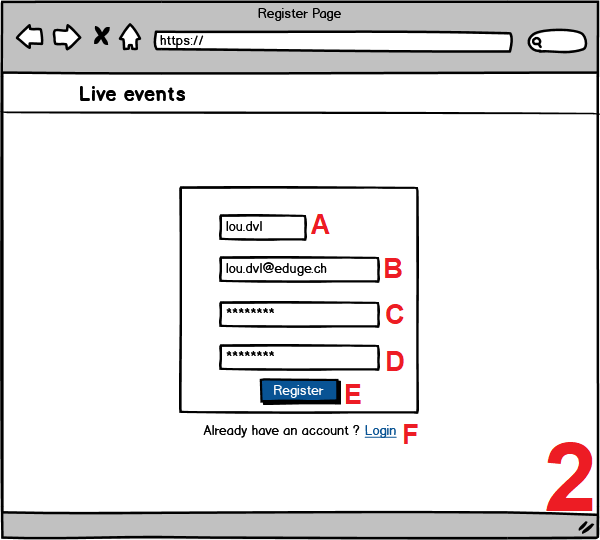


1. Le champ “Nickname” doit être rempli avec le pseudo de l’utilisateur
2. Le champ “Password” doit être rempli avec le mot de passe de l’utilisateur
3. Le bouton “Login” permet de confirmer la connexion à l’aide des information renseignées plus haut
4. Le lien “Sign up” redirige vers la page register.php permettant de se créer un compte utilisateur

### 

### 3.1.2 Enregistrement

La page permettant de de se créer un compte utilisateur afin d'accéder au reste des fonctionnalités du site. Si l’utilisateur possède un compte et veut directement se connecter il peut cliquer sur le lien “Login” afin d’être redirigé vers la page de connexion.



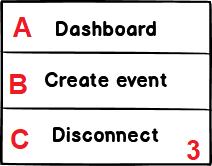
1. Le champ “Nickname” doit être rempli avec le pseudo souhaité par l’utilisateur
2. Le champ “E-mail” doit être rempli avec l’e-mail de l’utilisateur
3. Le champ “Password” doit être rempli avec le mot de passe souhaité par l’utilisateur
4. Le champ “Confirm Password” doit être rempli avec le même mot de passe que le point C afin de confirmer le mot de passe
5. Le bouton “Register” permet de confirmer la création du compte en envoyant les données entrées plus haut
6. Le lien “Login” permet à l’utilisateur d'accéder à la page login.php afin de se connecter si il a déjà un compte

## 

## 3.2 Partie utilisateur(site web)

### 3.2.1 Nav

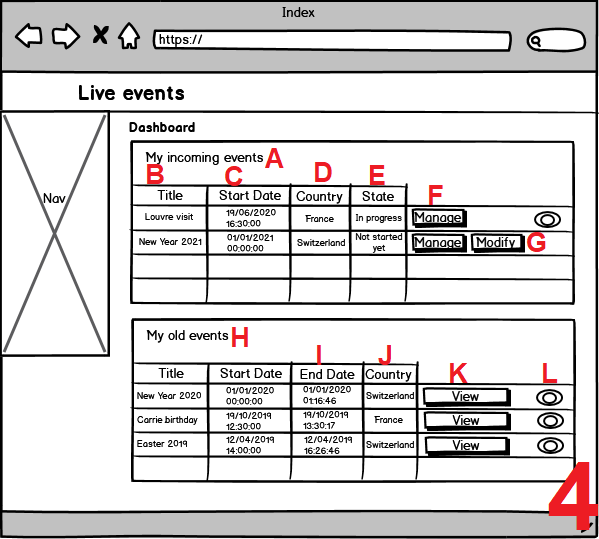
Le menu de navigation entre les différentes page du site web, il n’est présent que lorsque l’utilisateur est connecté.



1. Bouton redirigeant vers la page du tableau de bord affichant les événements de l’utilisateur
2. Bouton redirigeant vers la page de création d’événement
3. Bouton permettant de se déconnecter de la session en cours

### 3.2.2 Tableau de bord

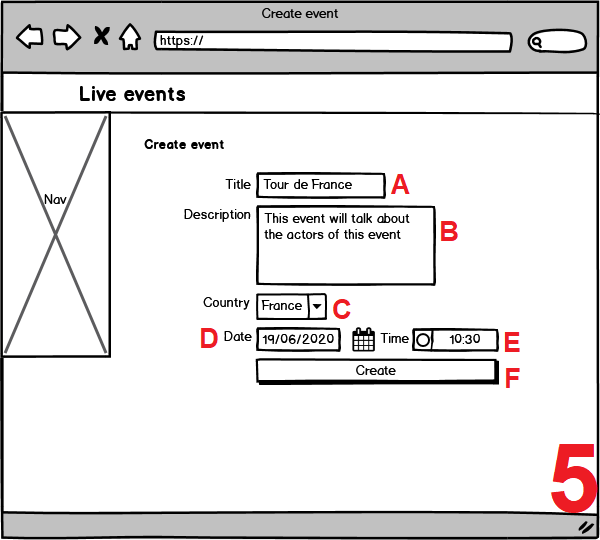
La page principale du site une fois connecté, elle permet de consulter les différents types d’événement que l’utilisateur a créé, il peut effectuer diverse action sur ces derniers : supprimer, accéder à la page de modification ou de gestion d’un événement.



1. La section “My incoming events” regroupe tous les événements en cours ou à venir créés par l’utilisateur
2. La colonne “Title” renseigne le titre de chaque événement
3. La colonne “Start Date” renseigne la date et l’heure de début de chaque événement
4. La colonne “Country” renseigne le pays attribué à chaque événement
5. La colonne “State” renseigne l’état actuel de chaque événement, il peut avoir 3 états différents dans cette section : “Not started yet”(Pas encore démarré), “In progress”(En cours)
6. Le bouton “Manage” redirige vers la page permettant de démarrer/stopper un événement et d’ajouter des infos au flux de ce dernier.
7. Le bouton modify redirige vers la page edit permettant de modifier les informations d’un événement
8. La section “My old events” regroupe les événements passés créés par l’utilisateur
9. La colonne “End date” renseigne la date et l’heure de fin de chaque événement passés
10. La colonne “Country” renseigne le pays attribué à chaque événement
11. Le bouton “View” redirige vers la page permettant de consulter un événement, son flux, de la masquer ou le supprimer
12. L'icône “Oeil” permet à l’utilisateur de voir si un événements est affiché ou masqué

### 3.2.3 Création d’événement

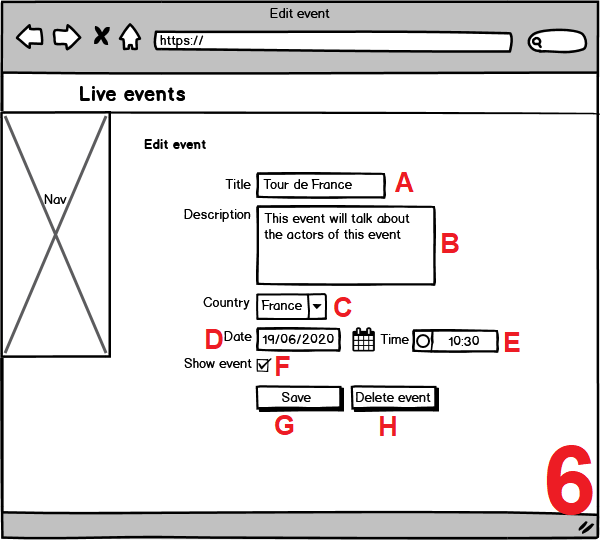
La page de création d’un événement, l’utilisateur y spécifie les diverse informations concernant l’événement et le créé. Une fois l’événement créé, l’utilisateur sera redirigé vers le dashboard où il pourra consulter ses événements(Écran 4).



1. Le champ “Title” doit être rempli avec le titre que l’utilisateur souhaite donner à son événement
2. Le champ “Description” doit être rempli avec la description de l’événement
3. Le dropdown “Country” permet de sélectionner le pays de l’événement
4. Le Datepicker “Date” permet de choisir la date de début de l’événement
5. Le Timepicker “Time” permet de choisir l’heure à laquelle l’événement commencera
6. Le bouton “Create” permet de confirmer l’envoi du formulaire rempli afin de créer l’événement

### 3.2.4 Modification d’événement

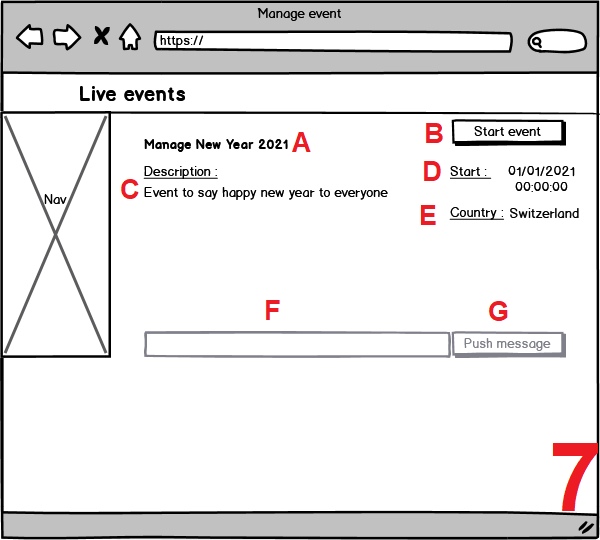
La page de modification d’un événement, l’utilisateur y spécifie les changements et valide le tout afin de mettre à jour les informations de l’événement.



1. Le champ “Title” doit être rempli avec le nouveau titre que l’utilisateur souhaite donner à son événement
2. Le champ “Description” doit être rempli avec la nouvelle description de l’événement
3. Le dropdown “Country” permet de sélectionner le nouveau pays de l’événement
4. Le Datepicker “Date” permet de choisir la nouvelle date de début de l’événement
5. Le Timepicker “Time” permet de choisir la nouvelle heure à laquelle l’événement commencera
6. La checkbox “Show event” peut être cochée ou décochée afin de cacher ou rendre visible l’événement pour les utilisateurs consultant les événements sur leur smartphone
7. Le bouton “Save” permet de confirmer l’envoi du formulaire rempli afin de mettre à jour les informations de l’événement
8. Le bouton “Delete event” permet de supprimer l’événement.

### 3.2.5 Gérer l’événement(Pas encore démarré)

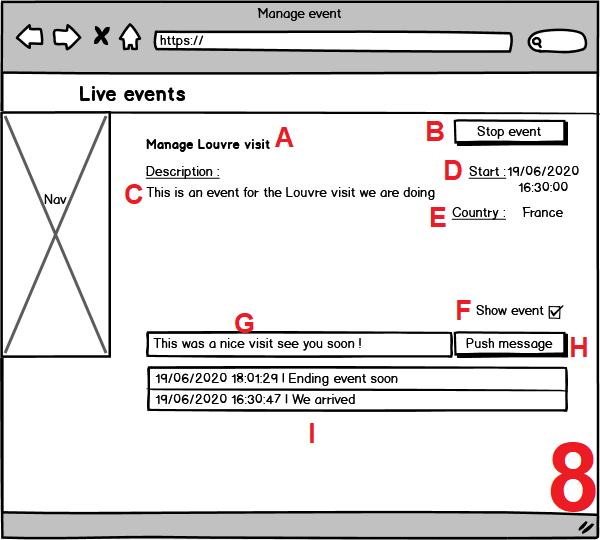
La page de gestion d’un événement, on y retrouve le flux de message de l’événement ainsi que la possibilité de commencer l’événement, dans cet état la page attend que l’utilisateur démarre l’événement avant de le laisser ajouter un message au flux de l’événement.



1. Le titre de la page sous la forme : “Manage” + Titre de l’événement
2. Le bouton “Start event” permet de démarrer l’événement, il passe son statut à : ”In progress”(En cours)
3. La description de l’événement
4. La date de début de l’événement
5. Le pays de l’événement
6. Champ permettant de rentrer un nouveau message à envoyer au flux de l’événement  
   il est à l’état “disabled” du moment que l’événement n’est pas lancé avec le bouton “Start event”.
7. Le bouton “Push message” permet de confirmer l’envoi du message entré dans le composant C afin d’ajouter un message au flux de l’événement, il est à l’état “disabled” du moment que l’événement n’a pas été lancé avec le composant “Start event”

### 3.2.6 Gérer l’événement(En cours)

La page de gestion d’un événement, on y retrouve le flux de message de l’événement ainsi que la possibilité d’arrêter l’événement ou de rajouter un message au flux.

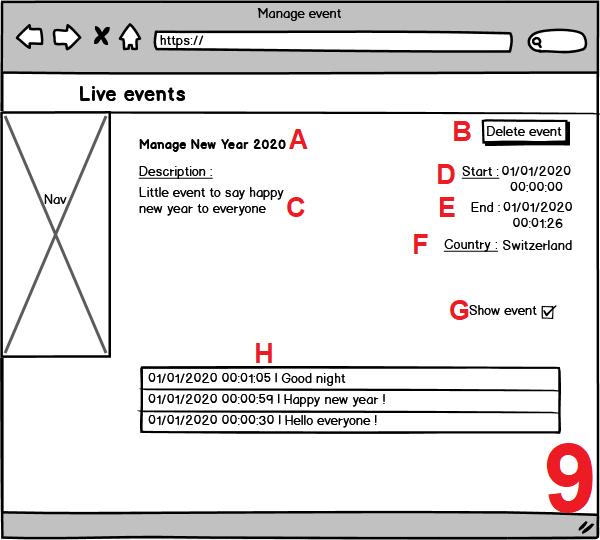


1. Le titre de la page sous la forme : “Manage” + Titre de l’événement
2. Le bouton “Stop event” permet de stopper l’événement en cours, il passe son statut à : ”Past”(Passé)
3. La description de l’événement
4. La date de début de l’événement
5. Le pays de l’événement
6. La checkbox “Show event” permettant de rendre visible ou invisible l’événement
7. Champ permettant de rentrer un nouveau message à envoyer au flux de l’événement.
8. Le bouton “Push message” permet de confirmer l’envoi du message entré dans le composant C afin d’ajouter un message au flux de l’événement
9. Affichage des messages du flux de l’événement sous la forme :  
   Date Heure | Message

### 

### 3.2.7 Gérer l’événement(Passé)

La page de gestion d’un événement, on y retrouve le flux de message de l’événement passé on ne peut ni redémarrer l’événement ni y ajouter de nouveaux message au flux.



1. Le titre de la page sous la forme : “Manage” + Titre de l’événement
2. Le bouton “Delete event” permet de supprimer l’événement passé
3. La description de l’événement
4. La date de début de l’événement
5. La date de fin de l’événement
6. Le pays de l’événement
7. La checkbox “Show event” permettant de rendre visible ou invisible l’événement
8. Affichage des messages du flux de l’événement sous la forme :  
   Date Heure | Message

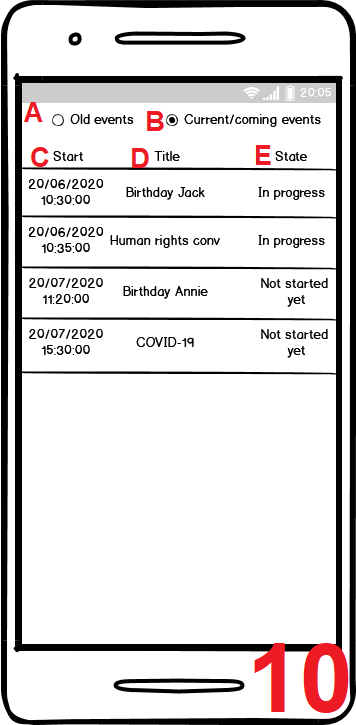
## 

## 3.3 Partie visiteur(Application Android)

### 3.3.1 Afficher événements En cours/Pas encore démarrés

L’écran principal de l’application android, accessible à tous, il permet de consulter tous les événements en cours ou futur, en jouant avec les radiobutton en haut de l’écran on peut afficher les événements en cours/futurs par une liste des événements terminés.

Si l’on souhaite consulter le flux ou les informations concernants un événement il suffit de cliquer dessus dans la liste.

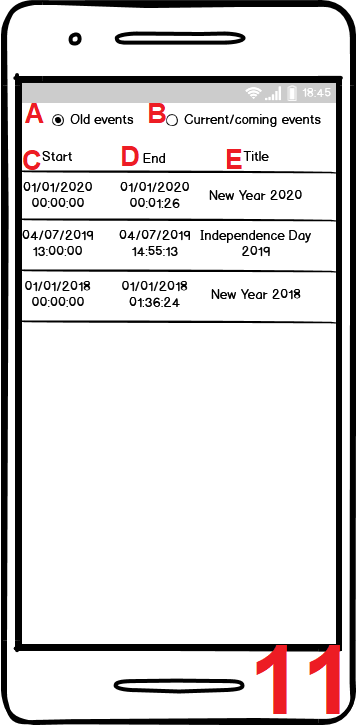


1. Le radiobutton “Old events” permet d’afficher tous les événements passés, il est lié au radiobutton “Current/coming events”(Composant B)
2. Le radiobutton “Current/coming events” permet d’afficher tous les événements en cours ou à venir, il est lié au radiobutton “Old events”(Composant A)
3. Label renseignant l’heure et la date de début de l’événement
4. Label renseignant le titre de l’événement
5. Label renseignant le statut de l’événement

### 3.3.2 Afficher événements Passés

L’écran principal de l’application android, accessible à tous, il permet de consulter tous les événements terminés, en jouant avec les radiobutton en haut de l’écran on peut afficher les événements en cours/futurs.

Si l’on souhaite consulter le flux ou les informations concernants un événement il suffit de cliquer dessus dans la liste.

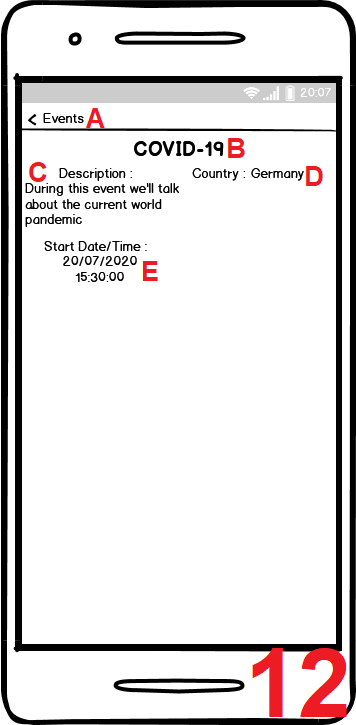


1. Le radiobutton “Old events” permet d’afficher tous les événements passés, il est lié au radiobutton “Current/coming events”(Composant B)
2. Le radiobutton “Current/coming events” permet d’afficher tous les événements en cours ou à venir, il est lié au radiobutton “Old events”(Composant A)
3. Label renseignant l’heure et la date de début de l’événement
4. Label renseignant l’heure et la date de fin de l’événement
5. Label renseignant le titre de l’événement

### 

### 3.3.3 Détails d’événement(Pas encore démarré)

On arrive sur cet écran lorsque l’on sélectionne un événement en pas encore démarré(Not started yet), on y retrouve toutes les informations relative à l’événement sélectionné.

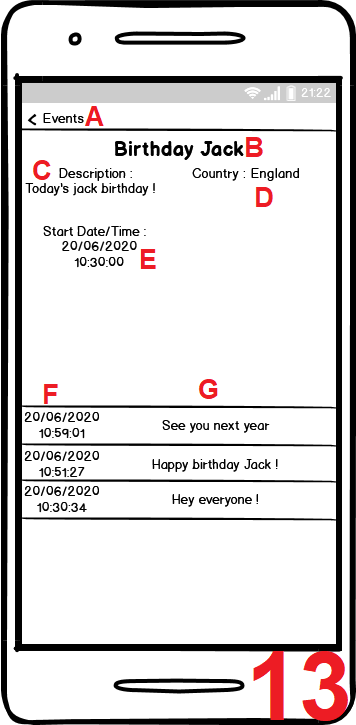


1. Le bouton “Events” permet de retourner à la page de consultation des événements
2. Label affichant le titre de l’événement sélectionné
3. Affichage de la description de l’événement sélectionné
4. Affichage du pays de l’événement sélectionné
5. Affichage de la date et de l’heure de début de l’événement sélectionné

### 

### 3.3.3 Détails d’événement(En cours)

On arrive sur cet écran lorsque l’on sélectionne un événement en cours(In progress), on y retrouve toutes les informations relative à l’événement sélectionné ainsi que le flux de l’événement.

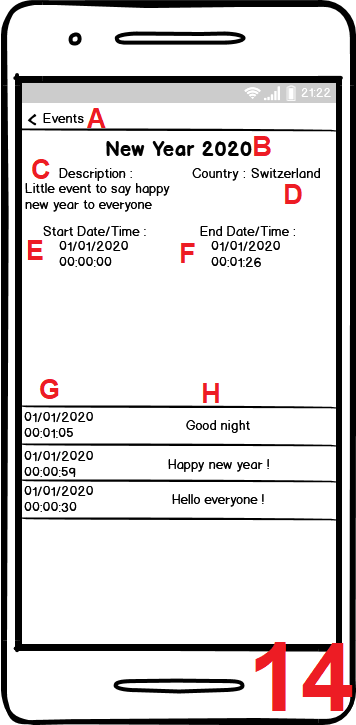


1. Le bouton “Events” permet de retourner à la page de consultation des événements
2. Label affichant le titre de l’événement sélectionné
3. Affichage de la description de l’événement sélectionné
4. Affichage du pays de l’événement sélectionné
5. Affichage de la date et de l’heure de début de l’événement sélectionné
6. Affichage de la date et de l’heure des messages du flux de l’événement sélectionné
7. Affichage des messages du flux de l’événement sélectionné

### 

### 3.3.3 Détails d’événement(Passé)

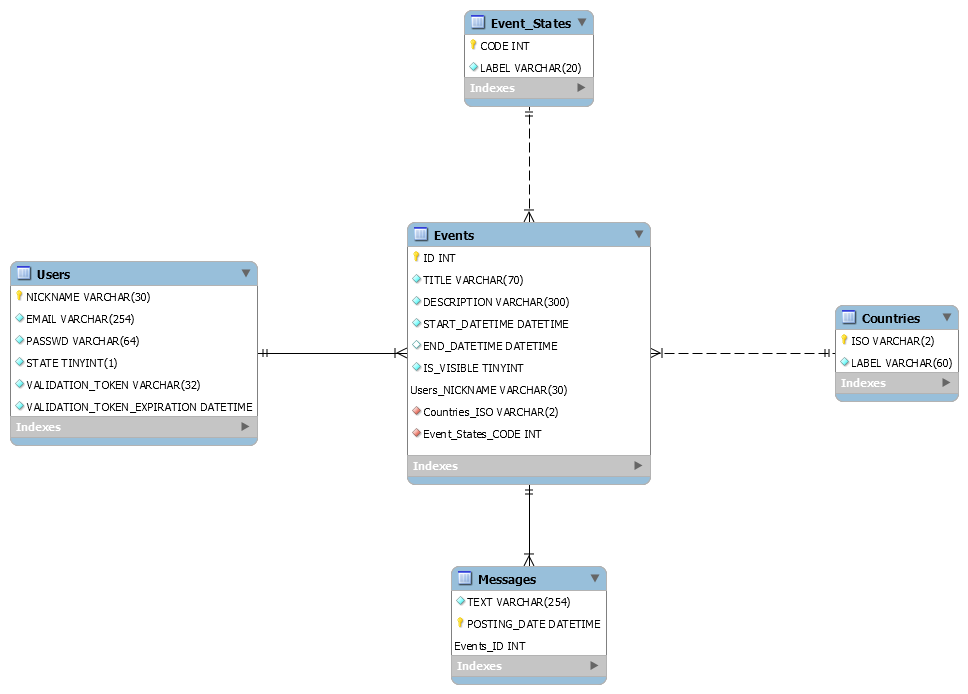
On arrive sur cet écran lorsque l’on sélectionne un événement passé(Past) , on retrouve sur cet écran toutes les informations relative à l’événement sélectionné ainsi que le flux de l’événement.



1. Le bouton “Events” permet de retourner à la page de consultation des événements
2. Label affichant le titre de l’événement sélectionné
3. Affichage de la description de l’événement sélectionné
4. Affichage du pays de l’événement sélectionné
5. Affichage de la date et de l’heure de début de l’événement sélectionné
6. Affichage de la date et de l’heure de fin de l’événement sélectionné si il en possède une
7. Affichage de la date et de l’heure des messages du flux de l’événement sélectionné
8. Affichage des messages du flux de l’événement sélectionné

# 4. Analyse organique

## 4.1 Modèle physique (Modèle relationnel)



### 4.1.1 Users

La table users contient les informations relatives aux utilisateurs.

**Champ Description Type Primary Key**

|  |  |  |  |
| --- | --- | --- | --- |
| NICKNAME | Le nom d’utilisateur | VARCHAR(30) | X |
| PASSWD | Le mot de passe de l’utilisateur | VARCHAR(64) |  |
| STATE | Le statut de l’utilisateur(0,1,2) | TINYINT(1) |  |
| VALIDATION\_TOKEN | La chaîne de caractères formant le token de validation par email | VARCHAR(32) |  |
| VALIDATION\_TOKEN\_EXPIRATION | La date d’expiration du token de confirmation | DATETIME |  |

### 4.1.2 Events

La table events contient les informations relatives aux événements des utilisateurs.

**Champ Description Type Primary Key**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | L’id de l’événement | INT | X |
| TITLE | Le titre de l’événement | VARCHAR(70) |  |
| DESCRIPTION | La description de l’événement | VARCHAR(300) |  |
| START\_DATETIME | La date de début de l’événement | DATETIME |  |
| END\_DATETIME | La date de fin de l’événement | DATETIME |  |
| STATE | L’état actuel de l’événement | ENUM(‘Not started yet’,’In progress’,’Past’) |  |
| IS\_VISIBLE | Défini si l’événement est caché ou visible | TINYINT |  |
| users\_NICKNAME | Le nom d’utilisateur de du créateur de l’événement | VARCHAR(30) |  |
| Countries\_ISO | Le code ISO 2 du pays de l’événement | VARCHAR(2) |  |

### 4.1.3 Countries

La table countries contient les différents pays et leur code ISO 2.

**Champ Description Type Primary Key**

|  |  |  |  |
| --- | --- | --- | --- |
| ISO | Le code ISO 2 du pays | VARCHAR(2) | X |
| LABEL | Le label du pays | VARCHAR(60) |  |

### 

### 4.1.4 Messages

La table messages contient les différents messages constituants les fluxs des événements.

**Champ Description Type Primary Key**

|  |  |  |  |
| --- | --- | --- | --- |
| TEXT | Le texte du message | VARCHAR(254) |  |
| POSTING\_DATE | La date et l’heure à laquelle le message a été posté | DATETIME | X |
| Events\_ID | L’id de l’événement auquel le message appartient | INT | X |

### 4.1.4 Event\_States

La table event\_states contient les différents statuts d’un événement ainsi que le code associé.

**Champ Description Type Primary Key**

|  |  |  |  |
| --- | --- | --- | --- |
| TEXT | Le texte du message | VARCHAR(254) |  |
| POSTING\_DATE | La date et l’heure à laquelle le message a été posté | DATETIME | X |
| Events\_ID | L’id de l’événement auquel le message appartient | INT | X |

## 4.2 Classes utilisées

### 4.2.1 Event

**La classe Event contient les informations associés à un événement.**

VARIABLES DESCRIPTION TYPE

|  |  |  |
| --- | --- | --- |
| id | L’id de l’événement | int |
| title | Le titre de l’événement | string |
| description | La description de l’événement | string |
| startDateTime | La date et l’heure de début de l’événement | string |
| endDateTime | La date et l’heure de fin de l’événement | string |
| state | L’état de l’événement | string |
| isVisible | Défini si l’événement est visible ou non | int |
| country | Le pays de l’événement | string |

### 

### 4.2.2 Message

**La classe Message contient les informations d’un message du flux.**

VARIABLES DESCRIPTION TYPE

|  |  |  |
| --- | --- | --- |
| text | Le texte du message | string |
| postingDate | La date et l’heure de publication du message | string |
| eventId | L’id de l’événement auquel appartient le message | int |

### 4.2.3 Country

**La classe Country contient les informations d’un pays.**

VARIABLES DESCRIPTION TYPE

|  |  |  |
| --- | --- | --- |
| iso | Le code iso-2 du pays | string |
| label | Le label contenant le nom du pays | string |

## 4.3 Manager utilisés

### 4.3.1 UserManager

**Classe contenant les différentes actions possibles avec un utilisateur**

#### connect($nickname,$passwd)

**Permet de vérifier les informations de connexion d’un utilisateur dans la base de données au moment du login.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| nickname | string | Le nom d’utilisateur |
| passwd | string | Le mot de passe de l’utilisateur |

Return : true si les 2 infos sont valides, false autrement

#### userExist($nickname)

**Vérifie si le nom d’utilisateur est déjà existant dans la base de données.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| nickname | string | Le nom d’utilisateur |

Return : true si le nom d’utilisateur existe sinon false

#### createUser($nickname,$email,$passwd)

**Crée un nouvel utilisateur dans la base de données**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| nickname | string | Le nom d’utilisateur |
| email | string | L’email de l’utilisateur |
| passwd | string | Le mot de passe de l’utilisateur |

Return : retourne le token de confirmation de l’utilisateur, retourne une chaîne vide sinon

#### validateUser($nickname,$token)

**Passe le statut d’un utilisateur à “vérifié”(1) dans la base de données**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| nickname | String | Le nom d’utilisateur |
| token | String | Le token de validation associé à l’utilisateur |

Return : true si l’utilisateur est validé

### 4.3.2 SessionManager

**Le SessionManager regroupe les différentes opérations possible avec la session.**

#### addNickname($nickname)

**Ajoute le nom d’utilisateur dans la session**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| nickname | String | Le nom d’utilisateur |

Return : Retourne true si l’ajout est réussi

#### getNickname()

**Récupère le nom d’utilisateur dans la session.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| - | - |  |

Return : Retourne un String contenant le nickname

#### destroySession()

**Détruit la session en cours.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| - | - | - |

Return : Retourne true si la session a bien étée détruite.

### 4.3.3 EmailManager

**L’EmailManager permet de gérer le module d’envoi de message.**

#### sendEmail($to,$subject,$text)

**Permet l’envoi d’un email**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| to | string | Le destinataire de l’email |
| subject | string | Le sujet de l’email |
| text | string | Le contenu de l’email |

Return : true si le mail a été envoyé

### 4.3.4 EventManager

**L’EventManager gère les différentes actions possible avec des événements.**

#### getAllVisibleEvents($filter)

**Récupère dans la base de données tous les événements visibles passés ou en cours/pas encore démarrés de la base suivant la valeur du filtre.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| filter | boolean | Optionnel, défini à false pour les événements passés, true pour les événements en cours ou futurs |

Return : Retourne un array<Event>

#### getUserEvents($nickname,$filter)

**Récupère tous les événements d’un utilisateur dans la base de données.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| nickname | string | Le nom d’utilisateur |
| filter | boolean | Optionnel, défini à false pour les événements passés, true pour les événements en cours ou futurs |

Return : Retourne un array<Event>

#### getMessages($eventId)

**Récupère tous les messages constituant le flux d’un événement précis.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| eventId | int | L’id de l’événement auquel les messages sont associés |

Return : Retourne un array<Message>

#### addMessage($message,$eventId,$nickname)

**Ajoute dans la base de données un message d’un flux.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| message | string | Le texte du message à ajouter |
| eventId | int | L’id de l’événement auquel le message est associé |
| nickname | string | Le nom d’utilisateur |

Return : Retourne true si le message est ajouté

#### createEvent($nickname,$title,$description,$country,$startDateTime)

**Crée un événement dans la base de données.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| nickname | string | Le nom d’utilisateur |
| title | string | Le titre de l’événement |
| description | string | La description de l’événement |
| country | string | Le pays de l’événement |
| startDateTime | string | La date et l’heure de l’événement |

Return : true si l’événement a pu être créé

#### 

#### updateEvent($eventId,$title,$description,$country,$startDateTime,$isVisible)

**Modifie un événement dans la base de données.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| eventId | int | L’id de l’événement |
| title | string | Le titre de l’événement |
| description | string | La description de l’événement |
| country | string | Le pays de l’événement |
| startDateTime | string | La date et l’heure de l’événement |
| isVisible | boolean | Défini si l’événement est visible ou non |

Return : true si l’événement a pu être mis à jour

#### getUserEvent($eventId,$nickname)

**Récupère un événement précis grâce à son id et du nom d’utilisateur de l’owner de l’événement.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| eventId | int | L’id de l’événement à récupérer |
| nickname | string | Le nom d’utilisateur de l’owner de l’événement |

Return : Retourne un objet Event

#### getVisibleEvent($eventId)

**Récupère un événement visible grâce à son id.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| eventId | int | L’id de l’événement à récupérer |

Return : Retourne un objet Event

#### updateEventState($eventId,$nickname,$state)

**Met à jour l’état d’un événement dans la base de données grâce à son id et du nom d’utilisateur de l’owner de l’événement.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| eventId | int | L’id de l’événement à récupérer |
| nickname | string | Le nom d’utilisateur de l’owner de l’événement |
| state | int | Le code de l’état à assigner à l’événement |

Return : Retourne true si l’update est un succès

#### getEvent($eventId,$nickname)

**Récupère un événement précis grâce à son id et du nom d’utilisateur de l’owner de l’événement.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| eventId | int | L’id de l’événement à récupérer |
| nickname | string | Le nom d’utilisateur de l’owner de l’événement |

Return : Retourne un objet Event

#### updateEventVisibility($eventId,$nickname,$isVisible)

**Met à jour l’état d’un événement dans la base de données grâce à son id et du nom d’utilisateur de l’owner de l’événement.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| eventId | int | L’id de l’événement à mettre à jour |
| nickname | string | Le nom d’utilisateur de l’owner de l’événement |
| isVisible | boolean | Si l’événement est visible ou non |

Return : Retourne true si l’update est un succès

#### deleteEvent($eventId,$nickname)

**Supprime un événement dans la base de données**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| eventId | int | L’id de l’événement à supprimer |
| nickname | string | Le nom d’utilisateur de l’owner de l’événement |

Return : Retourne true si la suppression est un succès

#### getEventEnd($eventId)

**Récupère la date de fin d’un événement en particulier**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| eventId | int | L’id de l’événement |

Return : Retourne une chaîne de caractères contenant la date de fin de l’événement.

### 4.3.5 CountryManager

**Le CountryManager contient les différentes actions pouvant être effectuées avec les pays de la base de données.**

#### getAllCountriesInfos()

**Récupère dans la base de données tous les labels des pays.**

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| - | - | - |

Return : Retourne un array<Country> contenants les informations de chaque pays

### 4.3.6 EventStateManager

**L’EventStateManager contient les actions possibles avec un le Statut d’un événement.**

#### getLabel($code)

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| code | int | Le code de l’état à récupérer |

Return : Retourne un string contenant le label de l’état

## 4.4 Autres fonctions

### 4.4.1 displayFunc.php

**Ce fichier regroupe des fonctions d’affichage en php**

#### genCountriesSelect($default)

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| default | string | Facultatif : le code ISO-2 d’un pays a sélectionner par defaut dans le Select généré |

Return : Retourne un string contenant le code html du select

#### displayDashboard($default)

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| $default | boolean | Défini à true si l’on souhaite afficher les événements pas encore commencés ou en cours. Défini à false si l’on souhaite afficher les événements passés |

Return : Retourne un string contenant le code html du tableau à afficher

### 4.4.2 jqueryUtils.js

**Ce fichier regroupe des fonctions permettant de gérer l’affichage de certaines pages ou bien des appels AJAX vers des scripts PHP, on utilise ici le framework JQuery.**

#### displayManageInterface(currentState,isVisible)

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| currentState | int | Le code d’état actuel de l’événement que l’on gère |
| isVisible | boolean | Est ce que l’événement que l’on gère actuellement est visible ou non |

#### displayManageButton(currentState)

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| currentState | int | Le code d’état actuel de l’événement que l’on gère |

#### displayVisibilityCheckbox(isChecked)

PARAMÈTRE(S) TYPE Description

|  |  |  |
| --- | --- | --- |
| isChecked | boolean | Est ce que la checkbox à afficher est cochée ou non |

#### startEvent()

|  |  |  |
| --- | --- | --- |
| - | - | - |

#### stopEvent()

|  |  |  |
| --- | --- | --- |
| - | - | - |

#### getEventEndDateTime()

|  |  |  |
| --- | --- | --- |
| - | - | - |

Return : Retourne un string contenant la date et l’heure de fin de l’événement demandé

#### updateEventVisibility()

|  |  |  |
| --- | --- | --- |
| - | - | - |

#### sendMessage()

|  |  |  |
| --- | --- | --- |
| - | - | - |

#### loadMessage()

|  |  |  |
| --- | --- | --- |
| - | - | - |

Return : Retourne un objet javascript contenant les messages de l’événement demandés.

#### displayMessages()

|  |  |  |
| --- | --- | --- |
| - | - | - |

## 4.5 Class Database

**La classe Database est la classe permettant la connexion à la base de données, on utilise ici le design pattern Singleton.**

### 4.5.1 Variable

VARIABLES TYPE DESCRIPTION

|  |  |  |
| --- | --- | --- |
| pdoInstance | PDO | Variable contenant l’instance PDO permettant l’accès à notre base de données |

#### 

### 4.5.2 Méthodes

Méthode Description Paramètres

|  |  |  |
| --- | --- | --- |
| private \_\_construct() | Constructeur de la classe, privé pour ne pas créer de nouvelle instance avec ‘=new Database();’ | - |
| private \_\_clone() | En privé pour qu’on ne puisse pas cloner l’instance | - |
| public static getInstance() | Retourne l’instance PDO ou en créée une avant si aucune instance. | - |
| final public static \_\_callStatic() | Est appelée lorsqu’on appelle des méthodes inaccessibles statiquement | chrMethod : Nom de la fonction appelée  arrArguments : Tableau d’arguments à passer à chrMethod |

## 4.6 Class (C#) APIConnector

**La classe APIConnector est une classe statique disposant de méthodes permettants de gérer les données extérieurs dont l’application a besoin pour fonctionner.**

### 4.6.1 Variable

VARIABLES TYPE DESCRIPTION

|  |  |  |
| --- | --- | --- |
| - | - | - |

#### 

### 4.6.2 Méthodes

Méthode Description Paramètres

|  |  |  |
| --- | --- | --- |
| getData(string scriptURL) | Permet d’effectuer un appel a une page de script sur le web afin de récupérer des données. Retourne une chaîne de caractères au format JSON contenant les données de la réponse de la page appelée | scriptURL : L’url de la page a appeler |
| decodeJWT(string data) | Décode une une chaîne de caractères encryptée et au format JWT | data : Une chaîne de caractères au format JWT |

## 4.7 Class (C#) Event

**La classe Event est une classe représentant un événement, elle contient toutes les informations concernant un événement.**

### 4.7.1 Variable

Propriétés TYPE DESCRIPTION

|  |  |  |
| --- | --- | --- |
| Id | int | L’id de l’événement |
| Title | string | Le titre de l’événement |
| Description | string | La description de l’événement |
| State | string | L’état de l’événement |
| Country | string | Le pays de l’événement |
| StartDateTime | DateTime | La date et l’heure de début de l’événement |
| EndDateTime | DateTime | La date et l’heure de fin de l’événement |
| Messages | List<Message> | Une liste contenant tous les messages du flux de l’événement |

#### 

### 4.7.2 Méthodes

Méthode Description Paramètres

|  |  |  |
| --- | --- | --- |
| - | - | - |

## 4.8 Class (C#) Message

**La classe Message est une classe représentant un message, elle contient toutes les informations concernant ce dernier.**

### 4.8.1 Variable

Propriétés TYPE DESCRIPTION

|  |  |  |
| --- | --- | --- |
| Text | string | Le texte du message |
| PostingDate | DateTime | La date et l’heure d’envoi du message |
| EventId | int | L’id de l’événement auquel est associé ce message |

#### 

### 4.8.2 Méthodes

Méthode Description Paramètres

|  |  |  |
| --- | --- | --- |
| - | - | - |

## 

## 4.9 Class (C#) EventsAdapter

**La classe EventsAdapter et une classe héritée de BaseAdapter, elle permet de créer des éléments dans la ListView contenant les événements et la remplit avec les valeurs des Events récupérés sur le serveur.**

### 4.9.1 Variable

Propriétés TYPE DESCRIPTION

|  |  |  |
| --- | --- | --- |
| \_items | List<Event> | Les éléments que l’on veut assigner à la ListView |
| \_context | Activity | Le context de l’application |
| \_filter | boolean | Si l’on veut afficher les événements pas encore commencés/en cours : true  Les événements passés : false |

#### 

### 4.9.2 Méthodes

Méthode Description Paramètres

|  |  |  |
| --- | --- | --- |
| EventsAdapter(Activity context, List<Event> items, bool filter) | Le constructeur de la classe | context : Le context de l’application en cours items : Les Messages à afficher  filter : Quel affichage l’on souhaite effectuer |
| long GetItemId(int position) | Retourne l’id de l’item à partir de sa position | position :la position de l’item dont l’on souhaite connaître l’id |
| View GetView(int position, View convertView, ViewGroup parent) | Crée remplit et retourne une ligne de la ListView | position : la position de l’élément à afficher convertView : l’ancienne vue créée à recycler si existante  parent : l’élément parent de la nouvelle vue |
| int Count | Retourne le nombre d'éléments présents dans la liste d’événement | - |
| Event this[int position] | Retourne un Event de la liste en fonction de la position donnée | position : la position de l’Event à chercher |

## 4.10 Class (C#) EventsAdapter

**La classe MessagesAdapter et une classe héritée de BaseAdapter, elle permet de créer des éléments dans la ListView contenant les messages d’un événement et la remplit avec les valeurs des Messages récupérés sur le serveur.**

### 4.10.1 Variable

Propriétés TYPE DESCRIPTION

|  |  |  |
| --- | --- | --- |
| \_items | List<Message> | Les éléments que l’on veut assigner à la ListView |
| \_context | Activity | Le context de l’application |

#### 

### 4.10.2 Méthodes

Méthode Description Paramètres

|  |  |  |
| --- | --- | --- |
| EventsAdapter(Activity context, List<Event> items) | Le constructeur de la classe | context : Le context de l’application en cours items : Les Messages à afficher |
| long GetItemId(int position) | Retourne l’id de l’item à partir de sa position | position :la position de l’item dont l’on souhaite connaître l’id |
| View GetView(int position, View convertView, ViewGroup parent) | Crée remplit et retourne une ligne de la ListView | position : la position de l’élément à afficher convertView : l’ancienne vue créée à recycler si existante  parent : l’élément parent de la nouvelle vue |
| int Count | Retourne le nombre d'éléments présents dans la liste de message | - |
| Message this[int position] | Retourne un Message de la liste en fonction de la position donnée | position : la position du message à chercher |

## 

# 5. Tests

# 6. Planning prévisionnel

# 7. Planning effectif

# 8. Conclusion

# 9. Glossaire