Dossier du projet Professionnel

APPLICATION MOBILE BDE LA PLATEFORME

PREVEDAN JONATHAN | TP CONCEPTEUR D'APPLICATION | juin 2021

SOMMAIRE

Table des matières

INTRODUCTION AU PROJET

Présentation personnelle

Je m'appelle Jonathan PREVEDAN, j'ai actuellement 20 ans et je suis étudiant au sein de La Plateforme.

Je suis à la fin de ma deuxième année d'étude au sein de la plateforme afin de valider un Titre Professionnel RNCP de niveau II Concepteur développeur d'application.

LISTE DES COMPETANCES DU REFERENCIEL

- **Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité**

  - Maquetter une application
  - Développer une interface utilisateur de type desktop
  - 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

- **Concevoir et développer la persistance des données en intégrants 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

- **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
  - Concevoir une application
  - Développer les 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

RESUME DU PROJET EN ANGLAIS

During my school year I had to carry out a mobile application as well as several other projects. I am a student at the Platform to validate a professional title. During this meet I will present my work of the year. A mobile application, I made for the Platformers in order to bring them together after so many months away from one another due to the health crisis.

CAHIER DES CHARGES

Objectif

1. Offrir une identité numérique aux étudiants de la Plateforme_.
2. Simplifier l'organisation et la gestion d'événements au sein de la Plateforme_.
3. Permettre aux étudiants de communiquer grâce à un chat
4. Faire une boîte à idées anonyme.
5. Gestion cantine étudiante.
6. Système de partage de médias et de connaissances.

Caractéristiques

Offrir une identité numérique aux étudiants de la Plateforme :

Offrir la possibilité aux étudiants de pouvoir s'identifier au sein d'une application propre à leur école, ainsi l'étudiant pourra participer à la vie commune au sein de l'école par le biais des caractéristiques précisées ultérieurement.

Simplifier l'organisation et la gestion d'événements au sein de la Plateforme :

Permettre aux membres du BDE de créer des événements, les élèves pourront accéder aux détails de l'événement, s'y inscrire s'ils le souhaitent en payant ou non une participation.

Permettre aux étudiants de communiquer grâce à un chat :

Afin d'internaliser les discussions numériques à travers les étudiants, ces derniers pourront ainsi échanger à travers des 'channels' au sein de l'application.

Faire une boîte à idée :

Permettre aux étudiants de soumettre des idées anonymement, et aux autres élèves de voter pour cette idée (upvote / downvote) anonymement également.

Gestion cantine étudiante :

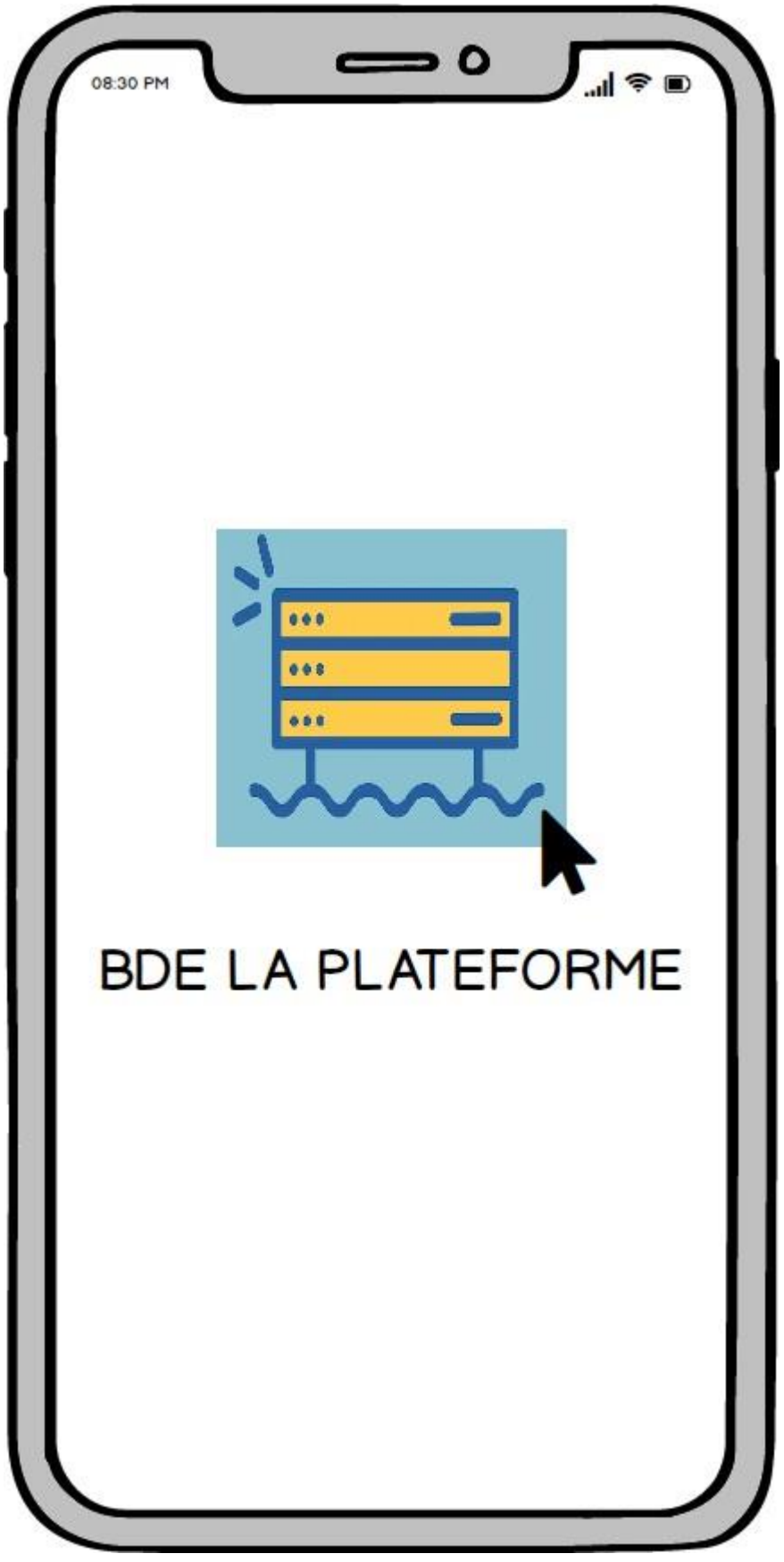
Aider les partenaires à vendre leurs produits en organisant un espace partenaire ou ils pourront y inscrire de nouveaux produits à mettre en vente, avec des réservations et des paiements gérés en dehors de l'application.

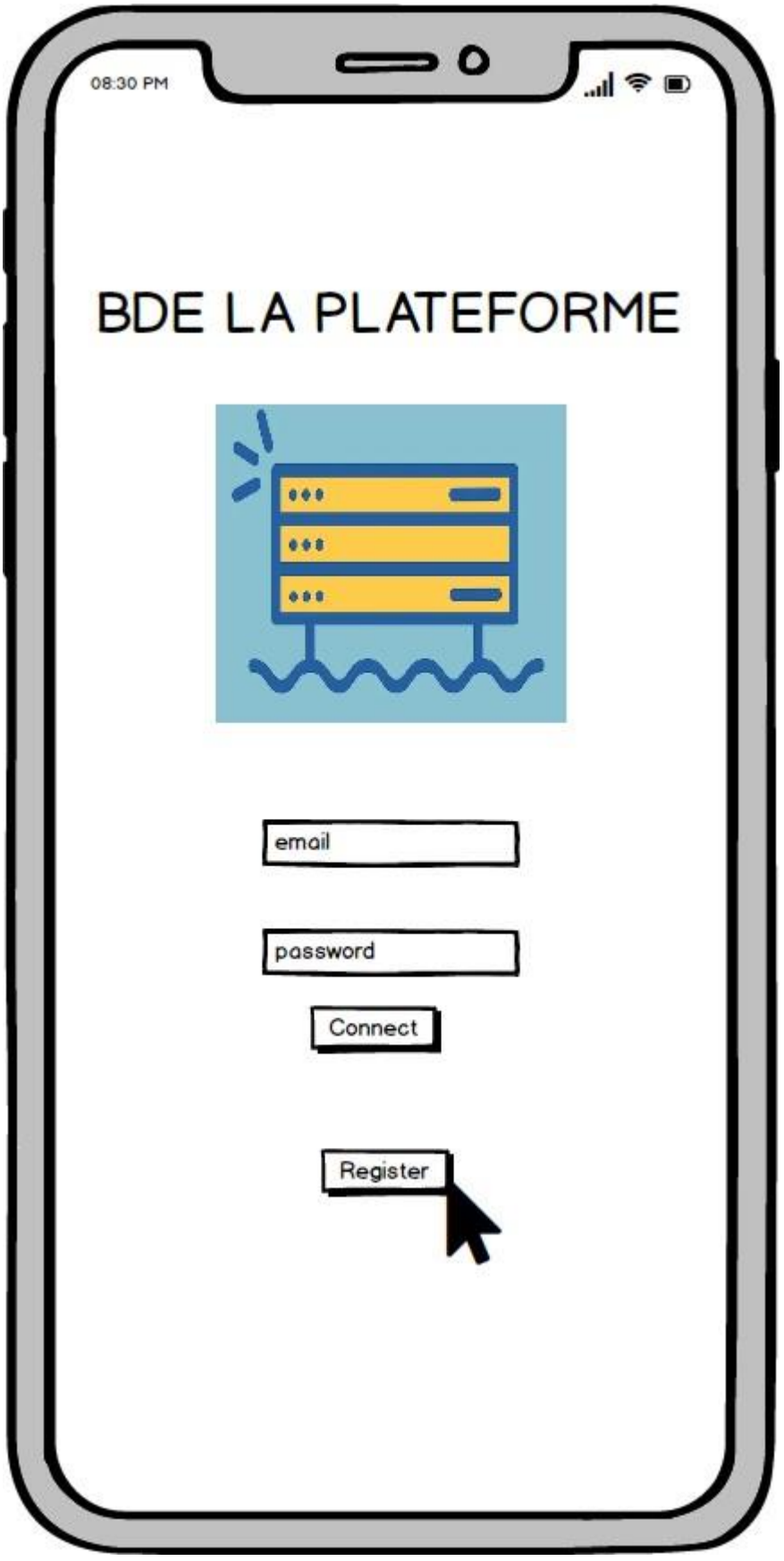
Système de partage de médias et de connaissances :

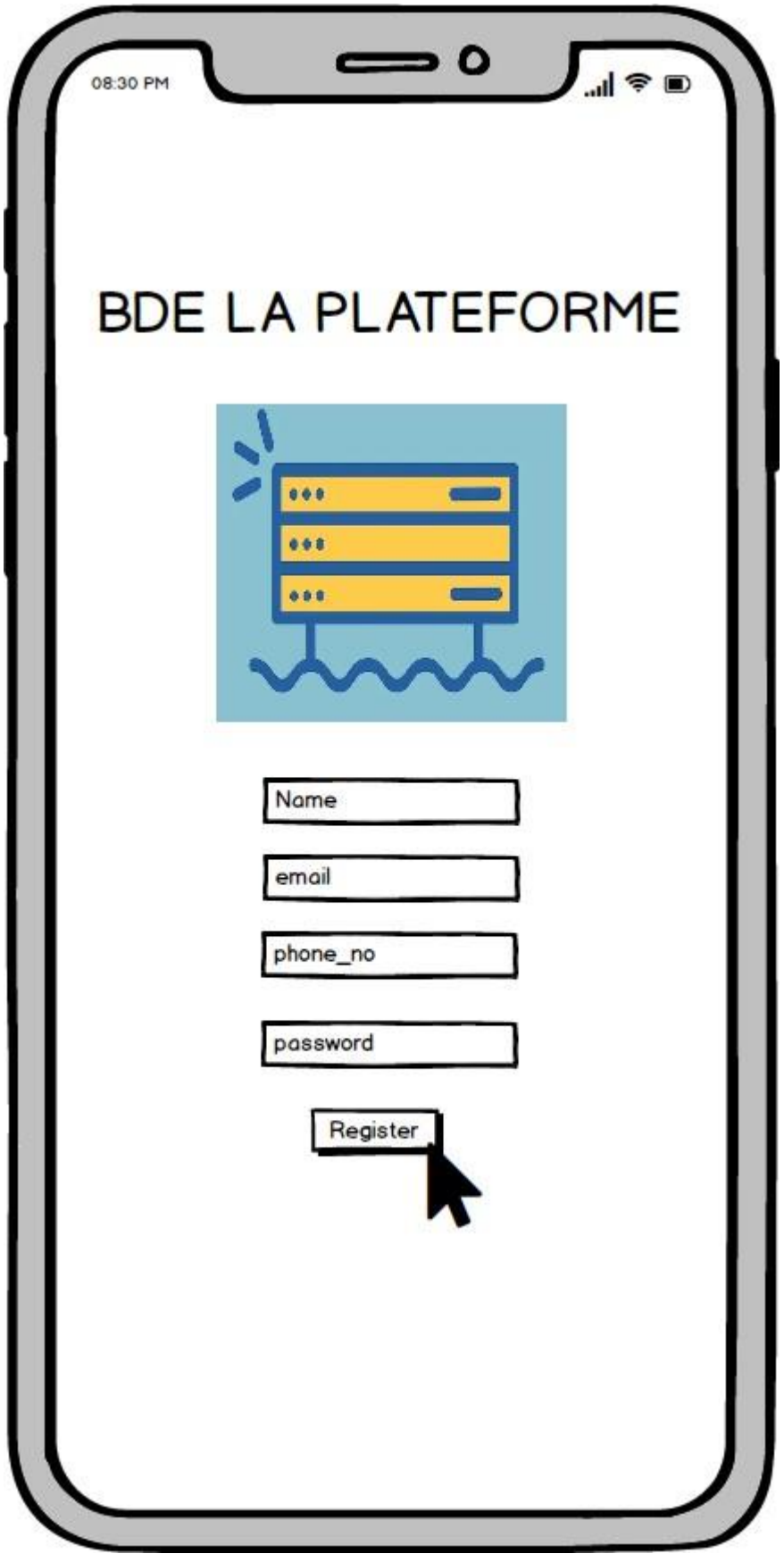
Elaborer un espace de partage pour les étudiants, afin de partager avec les autres des documentations techniques, des tutoriels

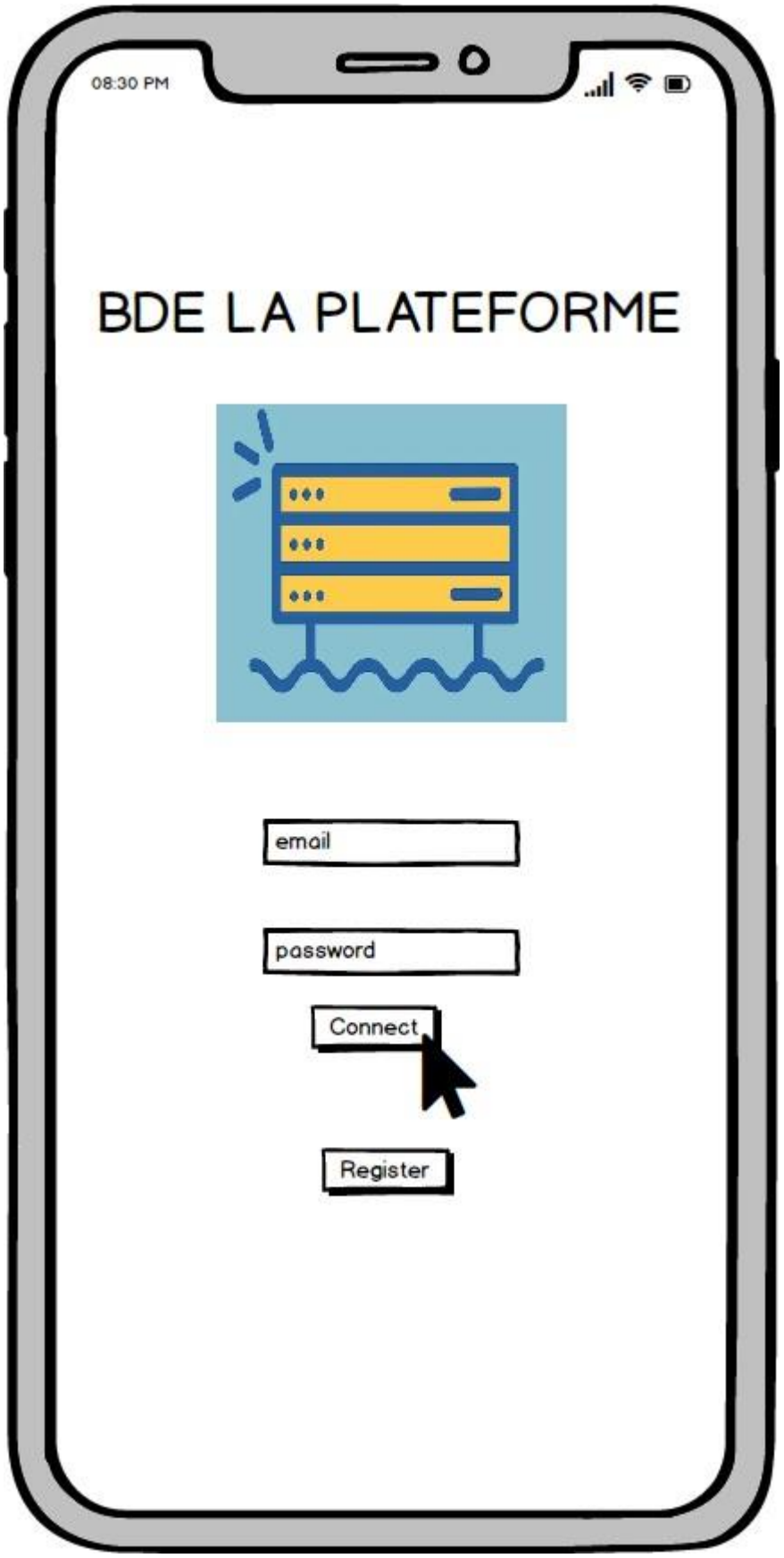
Système de OAUTH (Authentification Google)

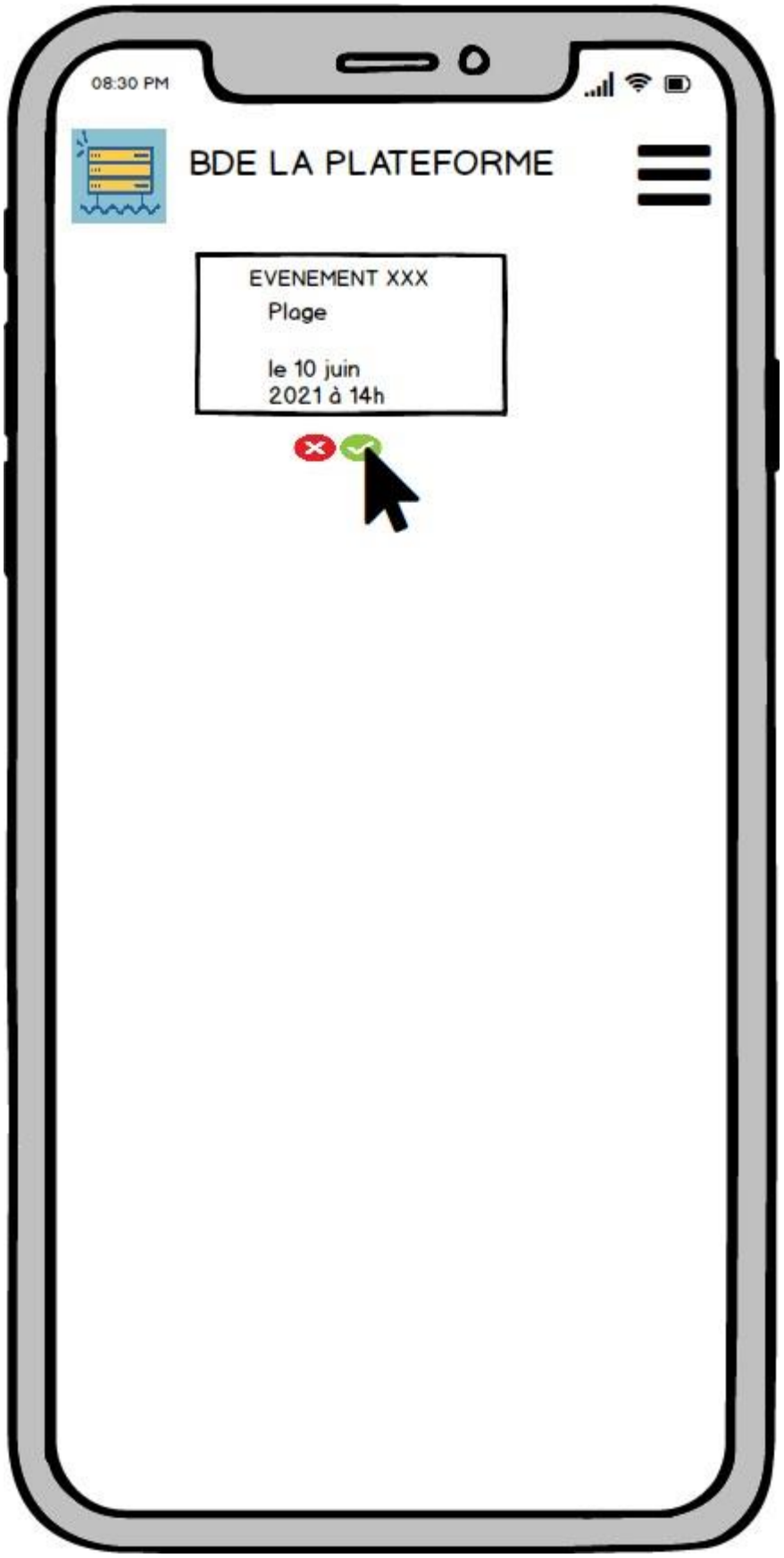
MAQUETTE DU PROJET

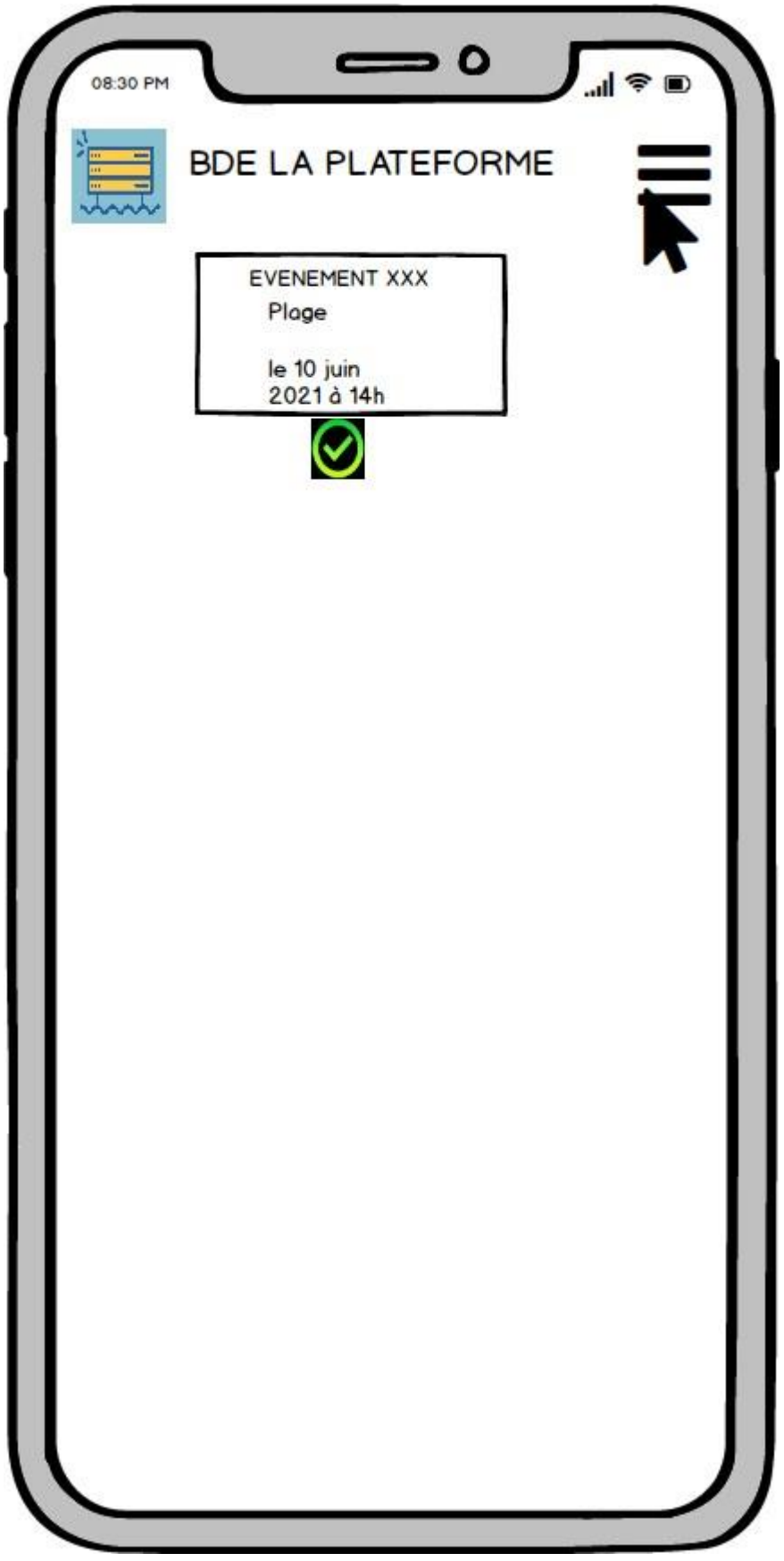

08:30 PM
BDE LA PLATEFORME


08:30 PM
BDE LA PLATEFORME
email
password
Connect
Register

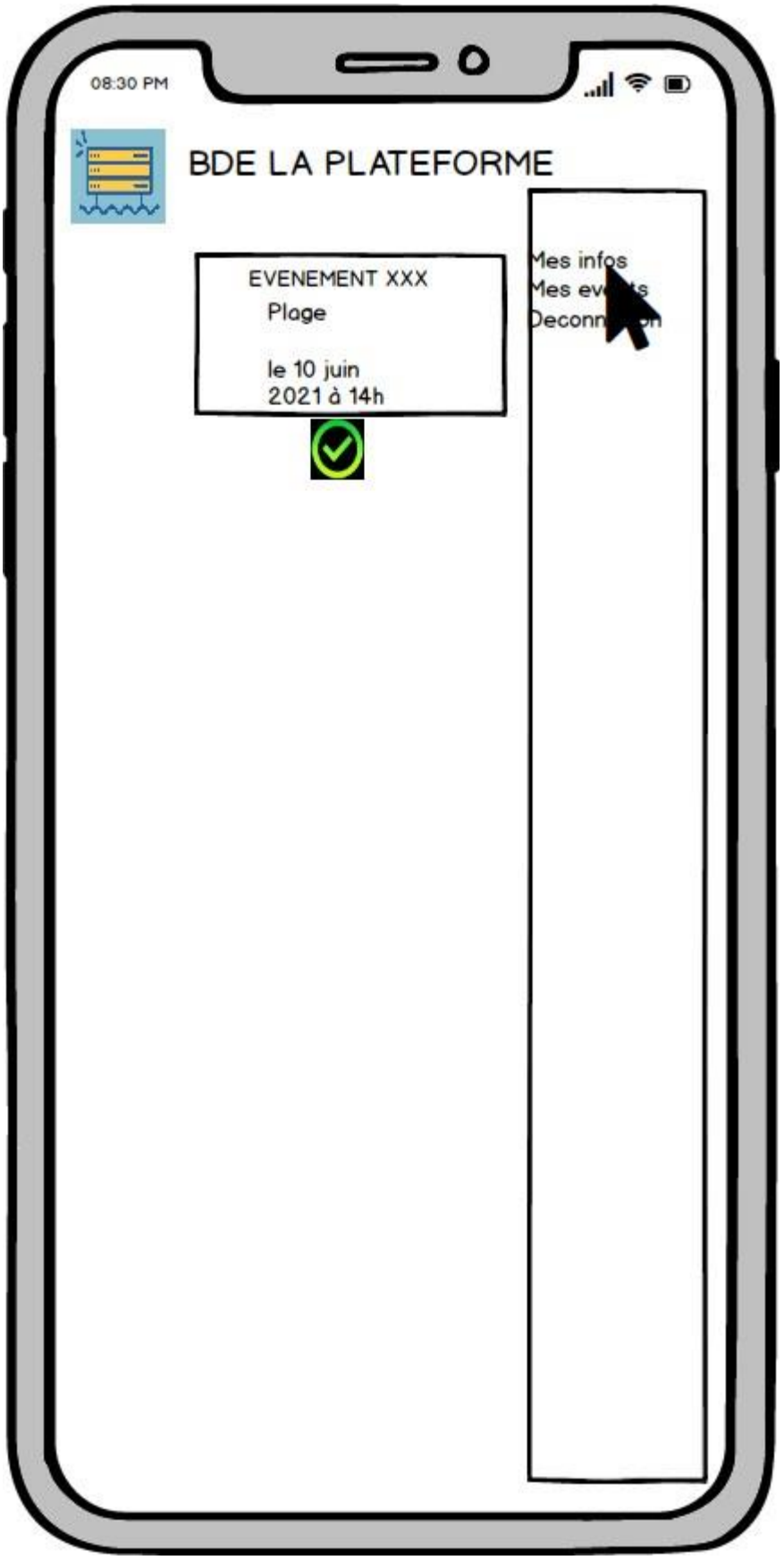

08:30 PM
BDE LA PLATEFORME
Name
email
phone_no
password
Register

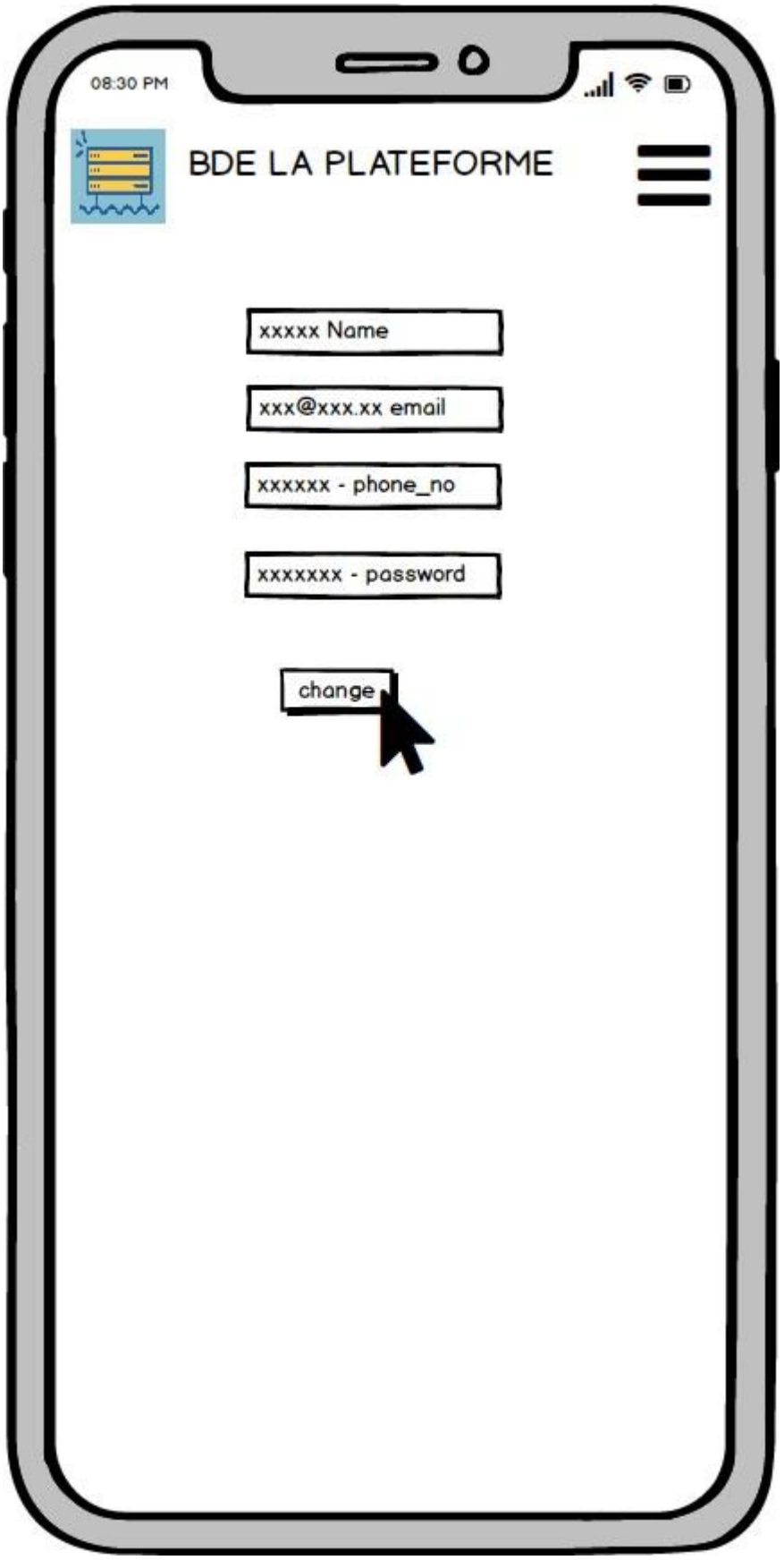

08:30 PM
BDE LA PLATEFORME
email
password
Connect
Register

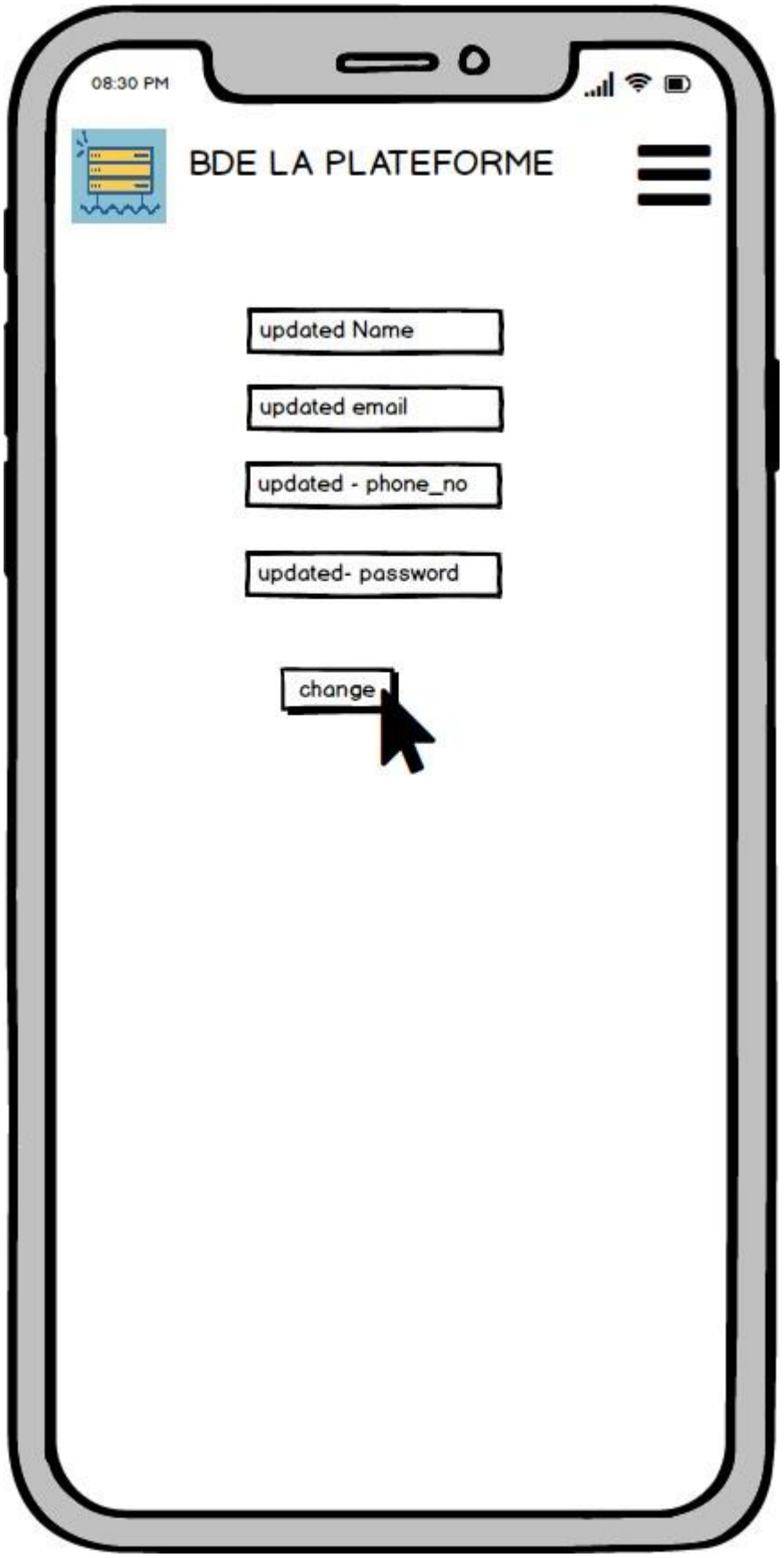

08:30 PM
BDE LA PLATEFORME
EVENEMENT XXX
Plage
le 10 juin
2021 à 14h

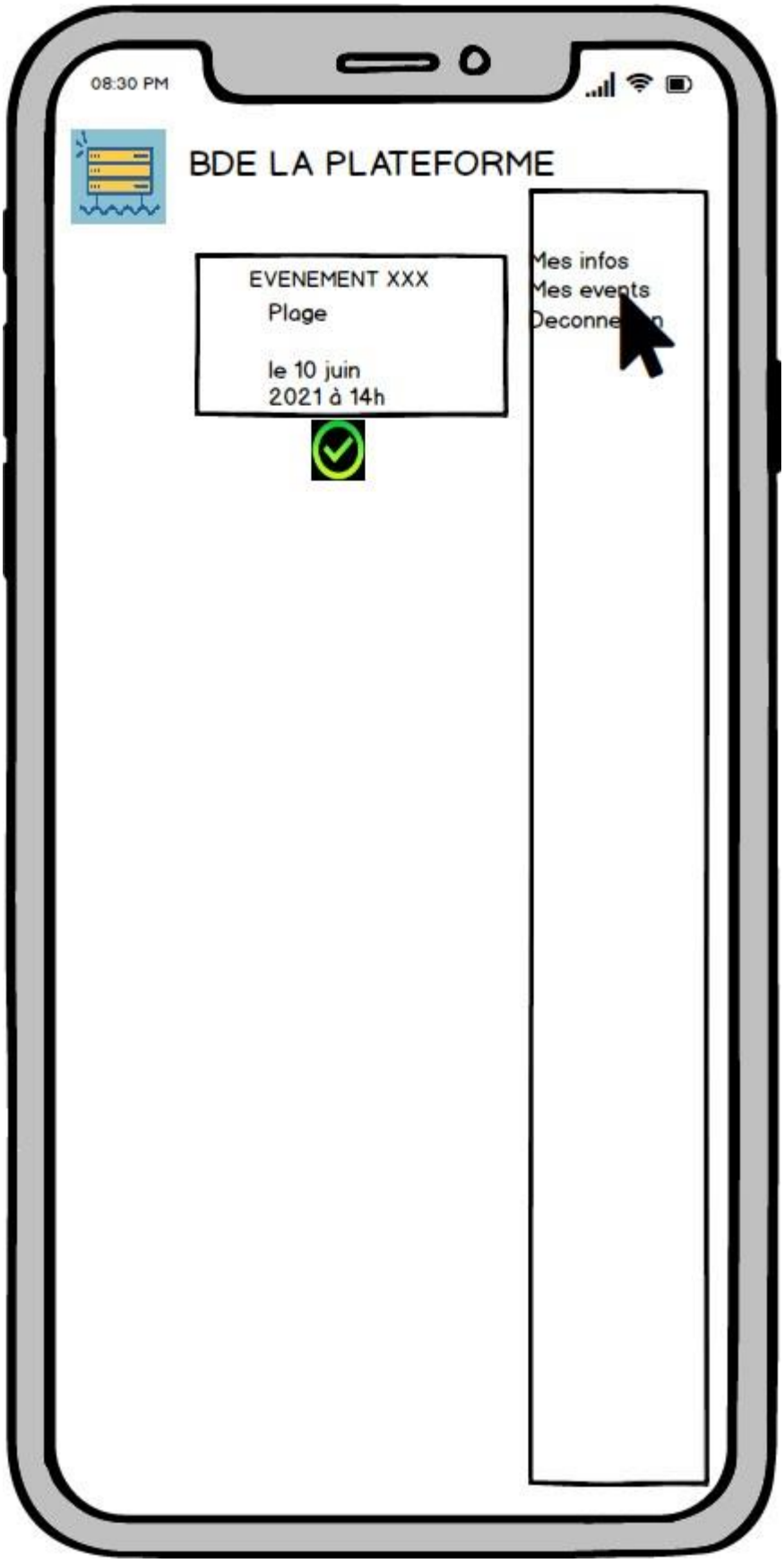

08:30 PM
BDE LA PLATEFORME
EVENEMENT XXX
Plage
le 10 juin
2021 à 14h

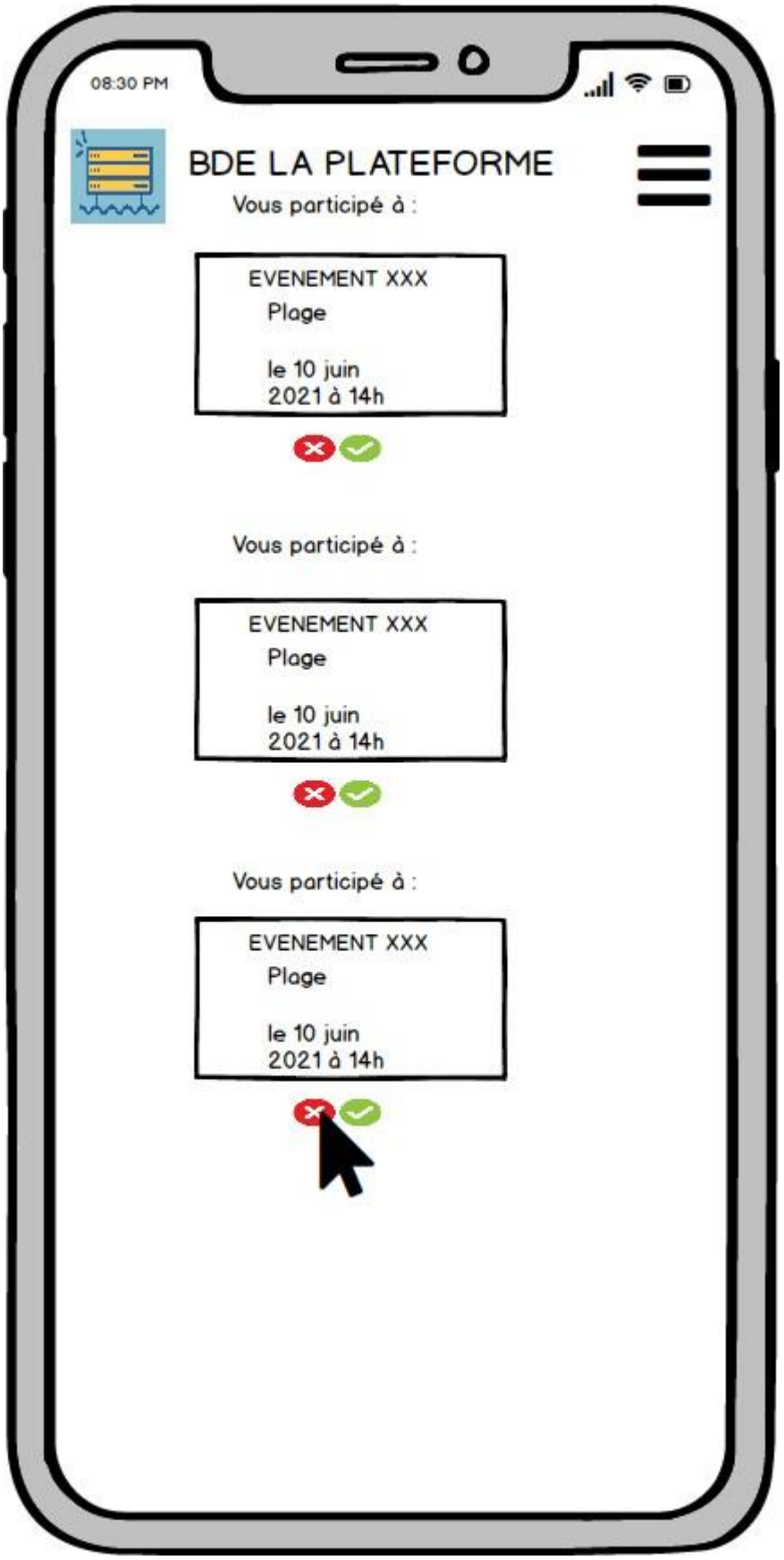
*

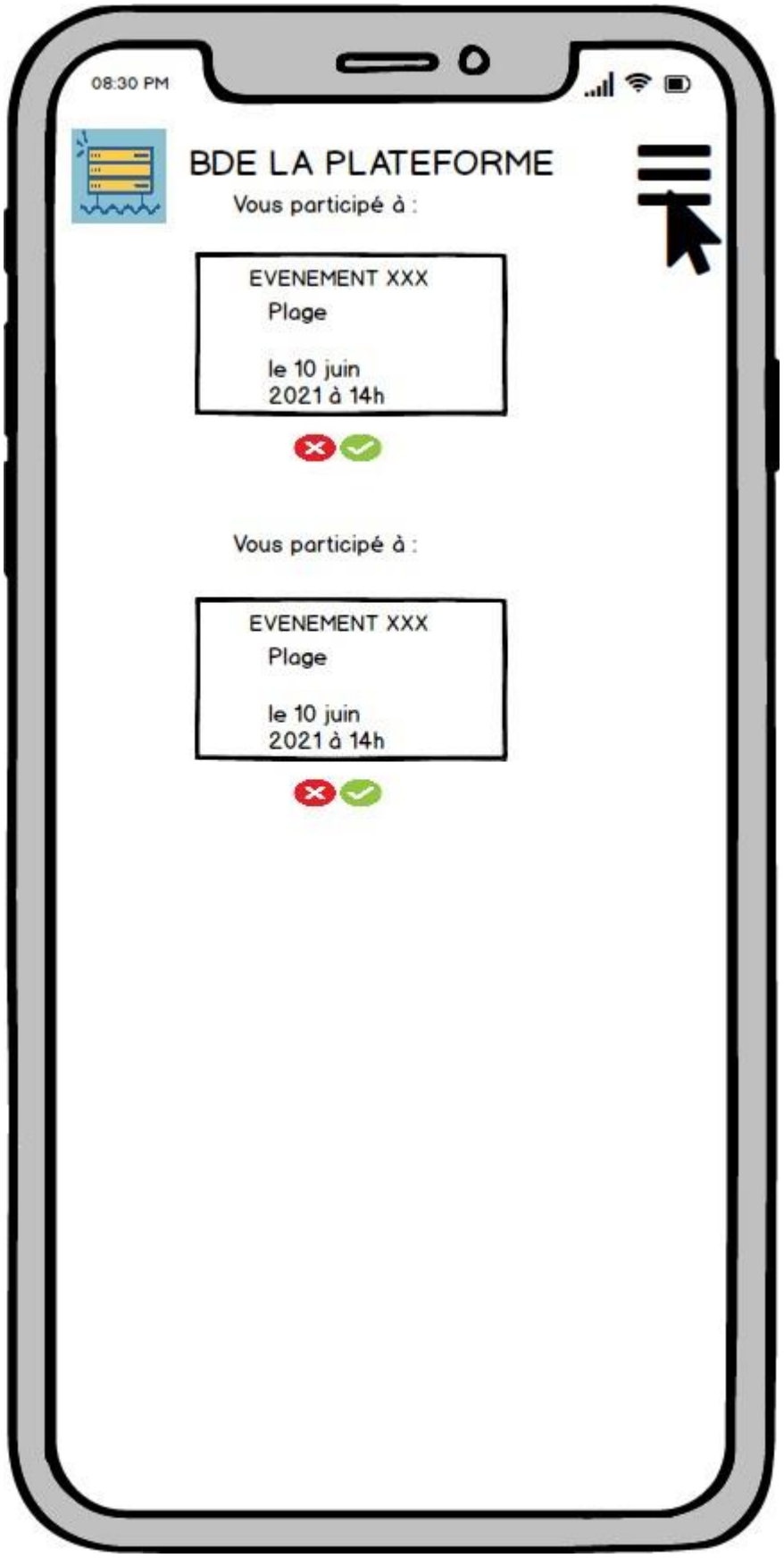

08:30 PM
BDE LA PLATEFORME
EVENEMENT XXX
Plage
le 10 juin
2021 à 14h
Mes infos


08:30 PM
BDE LA PLATEFORME
xxxxx Name
xxx@xxx.xx email
xxxxxx - phone_no
xxxxxxx - password
change


08:30 PM
BDE LA PLATEFORME
updated Name
updated email
updated - phone_no
updated- password
change


08:30 PM
BDE LA PLATEFORME
EVENEMENT XXX
Plage
le 10 juin
2021 à 14h
Mes infos
Mes events


08:30 PM
BDE LA PLATEFORME
Vous participé à :
EVENEMENT XXX
Plage
le 10 juin
2021 à 14h
Vous participé à :
EVENEMENT XXX
Plage
le 10 juin
2021 à 14h
Vous participé à :
EVENEMENT XXX
Plage
le 10 juin
2021 à 14h


08:30 PM
BDE LA PLATEFORME
Vous participé à :
EVENEMENT XXX
Plage
le 10 juin
2021 à 14h
Vous participé à :
EVENEMENT XXX
Plage
le 10 juin
2021 à 14h

Spécificités techniques du projet

Le projet est une application mobile.

Le mobile ne peut pas supporter de PHP, de ce fait tout ce qui transite des donnés doit être envoyé vers une URL via une requête HTTP.

De ce fait j'ai réalisé une API avec le Framework PHP CodeIgniter 4.

J'ai utilisé la librairie Firebase/JWT pour l'utilisation du Json Web Token, qui permet de crypter les informations liées à l'utilisateur (dans mon cas), via cette chaine de caractère crypté nous pouvons la décrypter et recevoir les informations qu'elle contient.

Concernant le développement du front, j'ai utilisé React Native

API hébergée sur mon hébergement web fournit par la plateforme Plesk.

Développement de l'API

Comme écris plus haut pour traiter des données une application mobile nécessite forcément de les envoyé à uns serveur back-end.

En l'occurrence dans mon projet mon API est en production sur mon hébergement web (fournit par la plateforme), mais pour y réaliser le développement, le serveur était en local (sous wamp)

Afin de réaliser mon API j'ai décidé de m'orienté sur le Framework PHP CodeIgniter, pourquoi ai-je choisis CodeIgniter ? Car pour moi je trouve ce Framework très puissant, je m'explique : Je trouve qu'il est assez simple à installer (installation via Composer une autre librairies PHP, dans mon cas), puis après quelques configurations comme la mise en place du mode = « development / production » et la mise en place de la connexion à la base de donnée, nous pouvons déjà commencer le développement de notre application. Ce Framework utilise le modèle MVC (Modèle Vue
Controller)

Modèle : type de donnée, objet.

Vue : Interface de l'utilisateur

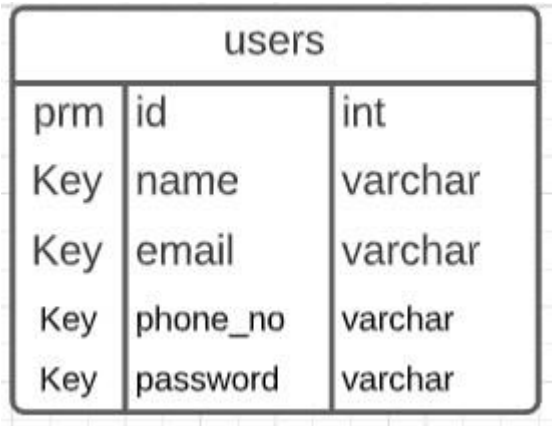
Controller : traitement des données, gestions des évènements.

De ce fait après avoir installé CodeIgniter dans mon dossier.

La question de la base de données nous rattrape.

Du MCD (Modèle conceptuel de données) au MPD (Modèle physique de données)

Entité users :



Au sein de cette entité 'users' nous retrouvons les champs que ma table users.

L'ID qui est la clé primaire de la table, qui correspond à l'ID unique de l'utilisateur. L'utilisateur ne peut avoir que 1 seul ID.

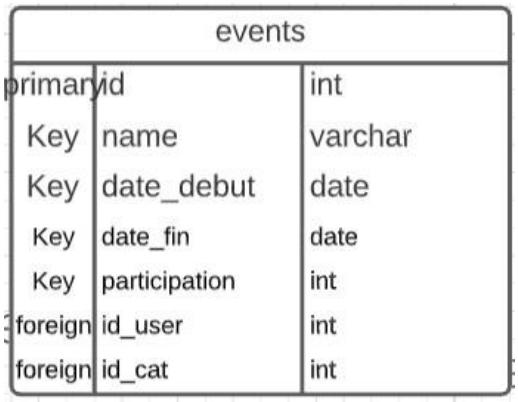
Name : type : varchar, correspond au nom de l'étudiant. Email : type :

varchar : correspond à l'email de l'étudiant

Phone_no : type : varchar : correspond au numéro de téléphone de l'étudiant.

Password : type : varchar : correspond au mot de passe crée par l'utilisateur à la création de son compte.

Entité Events :



Au sein de cette entité ‘events’ nous retrouvons les champs correspondant à ma table events de ma base de donnée.

La clé primaire est l’ID. L’événement ne peut avoir qu’un seul ID.

Name : type : varchar, correspondant au nom de l’evenement.

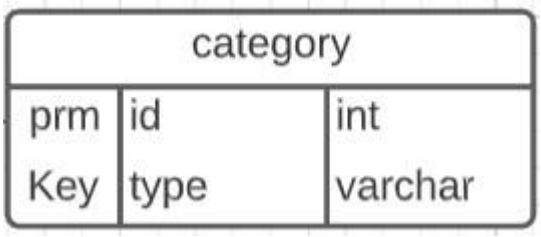
Date_debut : type : date, correspondant à la date de début de l’évènement, ce champ ne peut être null mais automatiquement au current_timestamp() si la date n’est pas renseigné.

Date_fin : type : date, correspondant à la date de fin de l’evenement, ce champ ne peut être null mais automatiquement au current_timestamp() si la date n’est pas renseigné.

Participation : type : int, correspondant à la participation de l’étudiant ou non.

Id_user : type : int clé étrangère, correspondant à l’ID unique de l’utilisateur, ce champ est une clé étrangère et est reliée à la table ‘users’ Id_cat : type : int clé étrangère, correspondant à l’ID unique de la catégorie, ce champ est une clé étrangère et est reliée à la table ‘catégorie’

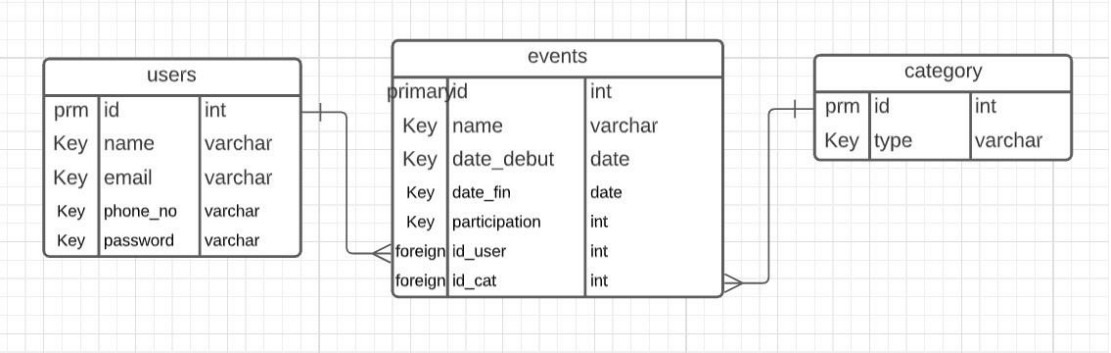
Entité ‘category’



La table category est bien mince, normal puisque je fais des clés étrangères dans ma table évènement qui contient le plus important de mon application.

Id : clé primaire int, unique ID de la catégorie

Type : varchar, correspondant au nom de la catégorie de l'évènement



Creation de la base de donnee

Après avoir effectué mon passe MCD à MPD, je peut enfin crée ma base de données comme il le faut.

Pour cela je lance mon serveur local (wamp) et me rend sur PhpMyAdmin.

Je crée donc ma base de donnée :

```
CREATE DATABASE API
```

Ceci fait, je me rend dans ma base de donnée API, puis sélectionne 'SQL' pour écrire en langage SQL la création de mes différentes tables.

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(120) DEFAULT NULL,
  `email` varchar(120) DEFAULT NULL,
  `phone_no` varchar(30) DEFAULT NULL,
  `password` varchar(120) DEFAULT NULL,
  `created_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=latin1;
```

Je crée donc ma table 'users' qui ressemble à ceci :

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	id	int(11)			Non	*Aucun(e)*		AUTO_INCREMENT	Modifier Supprimer Plus
2	name	varchar(120)	latin1_swedish_ci		Oui	*NULL*			Modifier Supprimer Plus
3	email	varchar(120)	latin1_swedish_ci		Oui	*NULL*			Modifier Supprimer Plus
4	phone_no	varchar(30)	latin1_swedish_ci		Oui	*NULL*			Modifier Supprimer Plus
5	password	varchar(120)	latin1_swedish_ci		Oui	*NULL*			Modifier Supprimer Plus
6	created_at	timestamp			Non	CURRENT_TIMESTAMP			Modifier Supprimer Plus

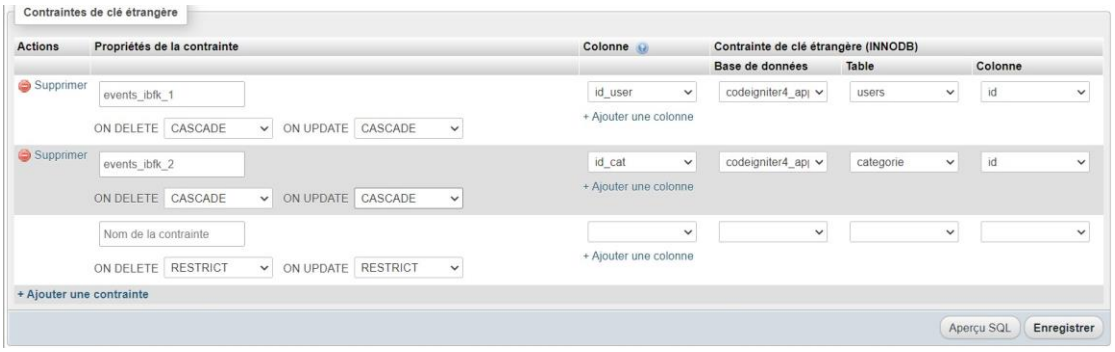
Je crée donc une autre table, ma table ‘events’ :

```
CREATE TABLE `events`(
`id` int(11) NOT NULL AUTO_INCREMENT,
 `name` varchar(120) DEFAULT NULL,
    `created_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `end_at` timestamp NOT NULL DEFAULT 0,
    `participation` int(11) DEFAULT NULL,
    `id_user` int(11) DEFAULT NULL,
    `id_cat` int(11) DEFAULT NULL,
    PRIMARY KEY (`id`),
    FOREIGN KEY (id_user) REFERENCES users(id),
    FOREIGN KEY (id_cat) REFERENCES categorie(id)
) ENGINE=INNODB AUTO_INCREMENT=18 DEFAULT CHARSET=latin1;
```

Qui ressemble à ceci :

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	id	int(11)			Non	*Aucun(e)*		AUTO_INCREMENT	Modifier Supprimer Plus
2	name	varchar(120)	latin1_swedish_ci		Oui	*NULL*			Modifier Supprimer Plus
3	created_at	timestamp			Non	CURRENT_TIMESTAMP			Modifier Supprimer Plus
4	end_at	timestamp			Non	0000-00-00 00:00:00			Modifier Supprimer Plus
5	participation	int(11)			Oui	*NULL*			Modifier Supprimer Plus
6	id_user	int(11)			Oui	*NULL*			Modifier Supprimer Plus
7	id_cat	int(11)			Oui	*NULL*			Modifier Supprimer Plus

Avec les clés étrangères :



Avec les suppressions et les updates en CASCADE = Supprimé l'évenements si ce dernier à été crée par un utilisateur qui à supprimé son compte.

Initialisation de CodeIgniter4

Une fois ma base de donnée crée, CodeIgniter installé dans mon dossier il faut configuré ce dernier pour qu'il puisse se connecte à la base de donnée.

Modification du fichier '.env'

Par défaut le fichier '.env' configure son environement en production, or pour le moment je le modifie pour le mettre en 'development' vu que je suis en local et encore en période de développement.

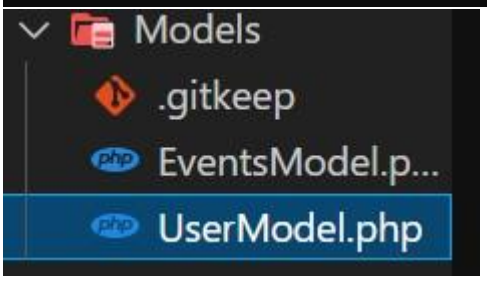
```
#--------------------------------------------------------------------
# DATABASE
#--------------------------------------------------------------------

 database.default.hostname = localhost
 database.default.database = codeigniter4_app
 database.default.username = jon
 database.default.password = jonathan
 database.default.DBDriver = MySQLi
database.default.DBPrefix =
```

Initialisation de la connexion à la base de donnée, en parcourant le fichier '.env' on établie les informations nécessaire à CodeIgniter4 pour effectuer la connexion à la base de donnée.

Models

Création des models via ligne de commande :

```
php spark make:model User --suffix
```



```php
class UserModel extends Model
{
    protected $DBGroup              = 'default';
    protected $table                = 'users';
    protected $primaryKey           = 'id';
    protected $useAutoIncrement     = true;
    protected $insertID             = 0;
    protected $returnType           = 'array';
    protected $useSoftDelete        = false;
    protected $protectFields        = true;
    protected $allowedFields        = [
        "name",
        "email",
        "phone_no",
        "password"
    ];

    // Dates
    protected $useTimestamps        = false;
    protected $dateFormat           = 'datetime';
    protected $createdField         = 'created_at';
    protected $updatedField         = 'updated_at';
    protected $deletedField         = 'deleted_at';
```

Ce model contient un tableau qui contient les champ possible de modifiés.

Controller :

```php
public function register()
{
    $rules = [
        "name" => "required",
        "email" => "required|valid_email|is_unique[users.email]|min_length[6]",
        "phone_no" => "required",
        "password" => "required",
    ];

    $messages = [
        "name" => [
            "required" => "Name is required"
        ],
        "email" => [
            "required" => "Email required",
            "valid_email" => "Email address is not in format"
        ],
        "phone_no" => [
            "required" => "Phone Number is required"
        ],
        "password" => [
            "required" => "password is required"
        ],
    ];

    if (!$this->validate($rules, $messages)) {

        $response = [
            'status' => 500,
            'error' => true,
            'message' => $this->validator->getErrors(),
            'data' => []
        ];
    } else {

        $userModel = new UserModel();

        $data = [
            "name" => $this->request->getVar("name"),
            "email" => $this->request->getVar("email"),
            "phone_no" => $this->request->getVar("phone_no"),
            "password" => password_hash($this->request->getVar("password"), PASSWORD_DEFAULT),
        ];
```

Les contrôleurs agissent comme une interface entre le modèle et la vue, pour traiter toute la logique métier et les requêtes entrantes, manipuler les données à l'aide du composant Modèle et interagir avec les Vues pour rendre le résultat final. Par exemple, le contrôleur « Client » va traiter toutes les interactions et les entrées de la Vue « Client » et mettre à jour la base de données en utilisant le Modèle « Client ». Le même contrôleur sera utilisé pour visualiser les données du client.

Ma fonction register permet d'enregister un utilisateur dans la base de donnée via une METHOD POST

J'établis premièrement un tableau contenant les conditions à respecté pour pouvoir s'inscrire dans la base de donnée.

Tous les champs sont requis, le champ email requière une adresse email valide, unique à l'utilisateur.

Ensuite je crée un tableau messages, ou je déclare les eventuels messages a retourné en cas de non-respect des conditions.

Je crée une première condition que si les conditions déclarais plus haut ne sont pas respectés alors on me retourne le message d'erreur.

Exemple :

Dans le cas ou l'utilisateur essaye de complété le formulaire en ne complétant aucune informations, ces messages d'erreurs lui sont retournés.

A l'inverse, si l'utilisateur remplit correctement le formulaire, alors ce dernier pourra s'inscrire dans la base de données.

```php
    if ($userModel->insert($data)) {

        $response = [
            'status' => 200,
            "error" => false,
            'messages' => 'Successfully, user has been registered',
            'data' => []
        ];
    } else {

        $response = [
            'status' => 500,
            "error" => true,
            'messages' => 'Failed to create user',
            'data' => []
        ];
    }
}
```

J'utilise la fonction insert() pour insérer mes données dans la base de données.

Si cela se passe correctement alors je lui retourne un message pour l'informer qu'il est bien inscrit.

A l'inverse je lui retourne ses informations et lui informe que l'enregistrement n'a pas fonctionné.

Une fois l'utilisateur enregistré, nous allons effectué la connexion.

```php
public function login()
{
    $rules = [
        "email" => "required|valid_email|min_length[6]",
        "password" => "required",
    ];

    $messages = [
        "email" => [
            "required" => "Email required",
            "valid_email" => "Email address is not in format"
        ],
        "password" => [
            "required" => "password is required"
        ],
    ];

    if (!$this->validate($rules, $messages)) {

        $response = [
            'status' => 500,
            'error' => true,
            'message' => $this->validator->getErrors(),
            'data' => []
        ];

        return $this->respondCreated($response);
```

Premièrement je crée un tableau ou j'établis les conditions à respectés.

Deuxièmement un autre tableau ou j'établis les messages à retourner en cas de non-respect des conditions préalablement déclarés.

Si l'utilisateur essaye de se connecter sans son adresse e-mail et/ou son mot de passe alors, je lui retourne le message d'erreur correspondant.

Exemple :


```php
else {
  $userModel = new UserModel();

  $userdata = $userModel->where("email", $this->request->getVar("email"))->first();

  if (!empty($userdata)) {

      if (password_verify($this->request->getVar("password"), $userdata['password'])) {

          $key = $this->getKey();

          $iat = time(); // current timestamp value
          $nbf = $iat + 10;
          $exp = $iat + 86400;

          $payload = array(
              "iss" => "The_claim",
              "aud" => "The_Aud",
              "iat" => $iat, // issued at
              "nbf" => $nbf, //not before in seconds
              "exp" => $exp, // expire time in seconds
              "data" => $userdata,
          );

          $token = JWT::encode($payload, $key);

          $response = [
              'status' => 200,
              'error' => false,
              'messages' => 'User logged In successfully',
              'data' => [
                  'token' => $token
              ]
          ];
          return $this->respondCreated($response);
```

A l'inverse, je crée une condition que si les champs des inputs ne sont pas vides alors :

Je vérifie le password car il est hash dans la base de donnée

Je fais appel à ma fonction getKey

```
private function getKey()
{
    return "my_application_secret";
}
```

Ensuite je récupère l'heure qu'il est actuellement et ajoute 86400 secondes qui représente 24 heures. Cela correspond à la durée de validité du token de connexion.

Au sein de la variable payload qui contient toutes les informations, l'heure de création, la durée de validité et les informations de l'utilisateur.

Je crée donc mon token grâce à JWT ::encode

Puis retourne le token en réponse à la connexion afin de pouvoir récupérer les informations de l'utilisateur.

Concernant le JWT j'utilise la librairie JWT/Firebase.

```
        return $this->respondCreated($response);
    } else {

        $response = [
            'status' => 500,
            'error' => true,
            'messages' => 'Incorrect details',
            'data' => []
        ];
        return $this->respondCreated($response);
    }
else {
    $response = [
        'status' => 500,
        'error' => true,
        'messages' => 'User not found',
        'data' => []
    ];
    return $this->respondCreated($response);
```

Si les informations de connexion ne se sont pas correctes, je retourne un message d'erreur. Si l'utilisateur n'est pas reconnu par son e-mail alors je lui retourne un message stipulant que aucun utilisateur n'a été trouvé.

Exemple :

Si un des champs est manquant :

Details :

```php
public function details()
{
    $key = $this->getKey();
    $authHeader = $this->request->getHeader("Authorization");
    $authHeader = $authHeader->getValue();
    $token = $authHeader;

    try {
        $decoded = JWT::decode($token, $key, array("HS256"));

        if ($decoded) {

            $response = [
                'status' => 200,
                'error' => false,
                'messages' => 'User details',
                'data' => [
                    'profile' => $decoded
                ]
            ];
            return $this->respondCreated($response);
        }
    } catch (Exception $ex) {

        $response = [
            'status' => 401,
            'error' => true,
            'messages' => 'Access denied',
            'data' => []
        ];
        return $this->respondCreated($response);
    }
}
```

Au sein de cette fonction je récupère le token qui se trouve dans
« Authorization » dans le header http

Je le decode grace à la fonction ::decode

Et je transmet les informations à l'utilisateur si les infos correspondent au token.

Update de donnes :

```php
public function update($id = null)
{
    $rules = [
        "name" => "required",
        "email" => "required|valid_email|min_length[6]",
        "phone_no" => "required",
        "password" => "required",
    ];

    $messages = [
        "name" => [
            "required" => "Name is required"
        ],
        "email" => [
            "required" => "Email required",
            "valid_email" => "Email address is not in format"
        ],
        "phone_no" => [
            "required" => "Phone Number is required"
        ],
        "password" => [
            "required" => "password is required"
        ],
    ];

    $model = new UserModel();
    $data = $model->find($id);


    $data = $this->request->getRawInput();
    if (!$this->validate($rules, $messages)) {
```

Dans mon tableau rules, j'établis les conditions qui doivent être rempli lorsque l'utilisateur souhaitera compléter les champs.

Ensuite je récupère la valeur des inputs, si les conditions ne sont pas remplies je retourne un message d'erreur.

A l'inverse alors j'UPDATE les valeurs dans la base de données.

Développement Front-End

Projet boutique en ligne

Pour réalise le côté front-end de ce site projet j'ai utilisé le language HTML5 CSS3 ainsi que JavaScript Vanilla (natif)

Développer une application type desktop

Projet : Tic Tac Toe

Afin de réaliser ce projet j'ai utilisé le langage Python ainsi que sa librairie PyGame

Le but de l'application est de réalisé le fameux jeu « morpion ou tic tac toe en anglais »

Une nouvelle s'ouvre à nous lorsque nous exécutons la page.

```
for event in pygame.event.get():

    if event.type == pygame.QUIT:
        sys.exit()
    # ajouter l'evenement qui correspond au clique droit
    if event.type == pygame.MOUSEBUTTONDOWN and pygame.mouse.get_pressed()[0]:

        # obtenir la position de la souris
        position = pygame.mouse.get_pos()
        #print(position)
        position_x ,position_y = position[0]//200 ,position[1]//200
        #print(self.position_x,self.position_y)

        # cond si le compteur est pair ou impair
        #print(self.compteur,self.compteur%2)
        if self.compteur % 2 == 0 :
            self.grille.fixer_la_valeur(position_x, position_y, self.player_X)


        else:
            self.grille.fixer_la_valeur(position_x, position_y, self.player_O)
        if self.grille.compteur_on:
            self.compteur += 1
            self.grille.compteur_on = False

        self.clicked = True

    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_RETURN:
            self.recommencer_le_jeu()
```

Ici nous ajoutons l'événement du clic droit. Nous obtenons la position de la souris afin de bien afficher le X ou le O au bon endroit de la fenêtre. Afin de savoir si c'est au joueur X ou O à jouer, j'instaure un compteur et compte le nombre de clique, si le clique est impair (3 par exemple alors ça sera le même signe que le premier à avoir jouer)

```
if len(liste_X) >= 3:
    for ligne, colonne in liste_X:
        liste_lignes_X.append(ligne)
        liste_colonnes_X.append(colonne)

    if liste_lignes_X.count(0) == 3 or liste_lignes_X.count(1) == 3 or liste_lignes_X.count(2) == 3:
        print('X WINS')

    if liste_colonnes_X.count(0) == 3 or liste_colonnes_X.count(1) == 3 or liste_colonnes_X.count(2) == 3:
        print('X WINS')

    if liste_lignes_X == liste_colonnes_X or liste_lignes_X == liste_colonnes_X[::-1]:
        print('X WINS')
```

Je compte les lignes et les colonnes, compare les résultats entre eux et affiche le X ou O vainqueur.

Conception d'une Application mobile

L'application est développée avec React Native et une intégration de Boostrap pour la gestion du style de l'application.

A VENIR…

Préparer et exécuter les plans de test d'une application

Afin de préparer mes plans de test, j'ai écrit plusieurs cas :

Cas	Attendu	Obtenu
Oublie de champ à l'enregistrement	Champ required	Champ required
Email invalide à l'inscription	Email adress is not in format	Email adress is not in format

Mes tests ont été réalisés en local via Postman

Postman sert à exécuter des appels HTTP directement depuis une interface graphique**.** Vous pourrez simplement choisir l'URL, la méthode HTTP *(le plus souvent GET, POST, PUT, PATCH et DELETE)*, les *headers*, les *query params* et dans certains cas le *body* de la requête.