



PRÉSENTATION DE PROJET

SONOW

VOTRE RÉSERVOIR D'ÉVÈNEMENTS

Soutenance Titre Professionnel Développeur web et web mobile

Par Frédéric GUIOU - Le 19 Octobre 2022



# SOMMAIRE

1. Présentation
2. Technologies & Outils Utilisés
3. Organisation & méthode de travail
4. Conception et production
5. Fonctionnalités représentatives
6. Jeu d'essai
7. Sécurisation de l'application
8. Résolution de problèmes rencontrés
9. Améliorations
10. Conclusion

# Présentation

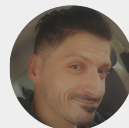
- À propos de moi
- Formation
- Contexte du projet
- Qu'est ce que SoNow ?





# Présentation

- À propos de moi
- Formation
- Contexte du projet
- Qu'est ce que SoNow ?

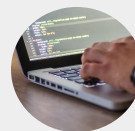


- 42 ans
- Formation initiale comptable (17 ans d'ex.)
- Passionné d'informatique et nouvelles technologies

SONOW

# Présentation

- ✓ À propos de moi
- Formation
- Contexte du projet
- Qu'est ce que SoNow ?



- Début de recherche en mi-2020
- Ecole O'clock
- Socle, Spé Data, Projet

SONOW

# Présentation

- ✓ À propos de moi
- ✓ Formation
- Contexte du projet
- Qu'est ce que SoNow ?



- Réintégration des notions
- Capacité de travail d'équipe
- Conditions professionnelles

SONOW

# Présentation

- ✓ À propos de moi
- ✓ Formation
- ✓ Contexte du projet
- Qu'est ce que SoNow ?



- Recherche d'événements (filtrage)
- Fonctionnalités Sociales (abonnements/abonnés)

SONOW

# Présentation

- ✓ À propos de moi
- ✓ Formation
- ✓ Contexte du projet
- ✓ Qu'est ce que SoNow ?



- Recherche d'événements (filtrage)
- Fonctionnalités Sociales (abonnements/abonnés)

SONOW





## Technologies et Outils Utilisés

- Technologies Front-end
- Technologies Back-end
- Outils de développement



REACT + REDUX



Axios



Semantic UI

## Technologies et Outils Utilisés

- Technologies Front-end
- Technologies Back-end
- Outils de développement

SONOW



REACT + REDUX



Axios



Semantic UI



Nodejs



Express Js



PostgreSQL



Sqitch

## Technologies et Outils Utilisés

- ✔ Technologies Front-end
- Technologies Back-end
- Outils de développement

SONOW



REACT + REDUX



Axios



Semantic UI



Nodejs



Express Js



PostgreSQL



Sqitch



GitHub



VS Code



Postman

## Technologies et Outils Utilisés

✓ Technologies Front-end

✓ Technologies Back-end

● Outils de développement

SONOW



REACT + REDUX



Axios



Semantic UI



Nodejs



Express Js



PostgreSQL



Sqitch



GitHub



VS Code



Postman

## Technologies et Outils Utilisés

✓ Technologies Front-end

✓ Technologies Back-end

✓ Outils de développement

SONOW



# Organisation & Méthode de travail

- 4 Sprints (1 semaine chacun)
- Déroulement d'une journée de travail



# Organisation & Méthode de travail

- 4 Sprints (1 semaine chacun)
- Déroulement d'une journée de travail

## MÉTHODE AGILE SCRUM

### -Sprint 0-

Conception théorique et documentation

### -Sprint 1-

Architecture et fonctionnalités CRUD

### -Sprint 2-

Fonctionnalités "sociales"

### -Sprint 3-

Finalisation et préparation de la présentation

SONOW

# Organisation & Méthode de travail

- ✓ 4 Sprints (1 semaine chacun)
- Déroulement d'une journée de travail

## Journée de travail type

- Réunion "Kick Off" 9h00
- Etat des "accomplis / à accomplir"
- Développement Single ou pair programming
- Point Mi-journée / Fin de journée

# Organisation & Méthode de travail

- ✓ 4 Sprints (1 semaine chacun)
- ✓ Déroulement d'une journée de travail

## Journée de travail type

- Réunion "Kick Off" 9h00
- Etat des "accomplis / à accomplir"
- Développement Single ou pair programming
- Point Mi-journée / Fin de journée



## Conception et production

- Détail des fonctionnalités
- Workflow de l'application
- Charte graphique
- Les routes
- MCD / MLD / Dico des données / MPD
- Wireframes



## User Stories

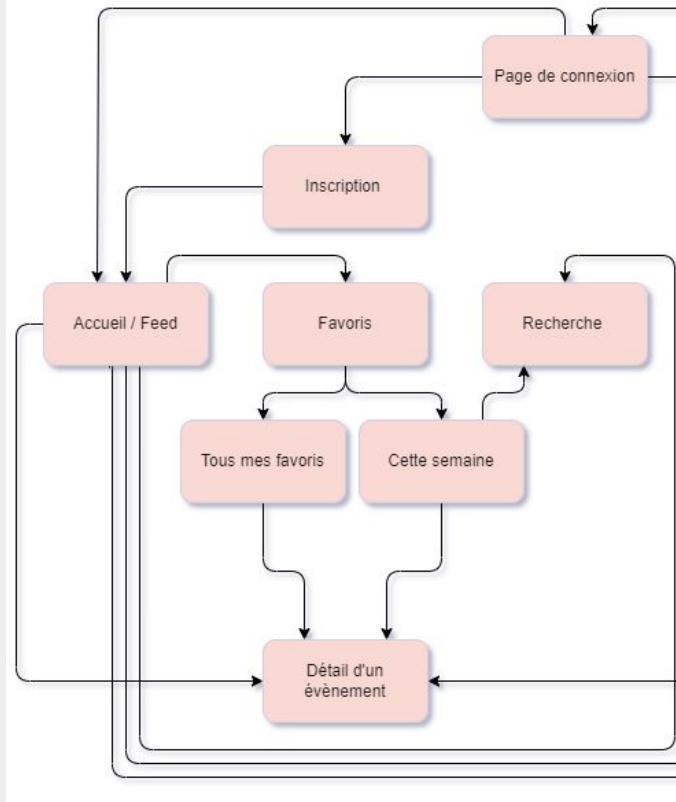
"En tant que..."	"J'ai besoin de..."	"Afin de..."
Visiteur	M'inscrire	D'utiliser l'application
Visiteur	Me connecter	D'utiliser l'application & accéder à mon compte utilisateur

"En tant que..."	"J'ai besoin de..."	"Afin de..."
utilisateur connecté	visualiser les événements du fil d'actualité	m'inspirer et trouver une activité
utilisateur connecté	accéder à la page de recherche des événements	rechercher un événement précis par son nom ou sa catégorie
utilisateur connecté	accéder à mes événements favoris	pour connaître leur actualité
utilisateur connecté	mentionner ma participation à un événement	informer la communauté de Sonowistes
utilisateur connecté	accéder à mon profil	le modifier/me déconnecter
...	...	...

## Conception et production

- Détail des fonctionnalités
- Workflow de l'application
- Charte graphique
- Les routes
- MCD / MLD / Dico des données / MPD
- Wireframes

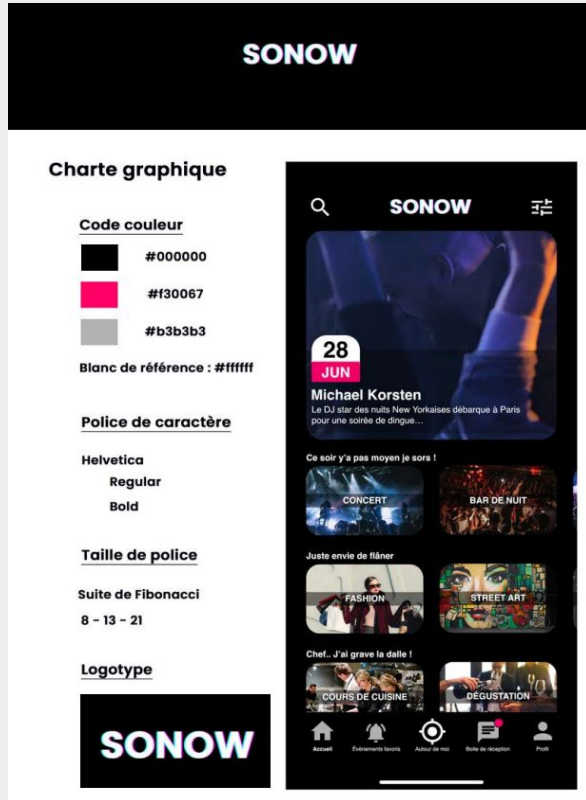
## Extrait du workflow de l'application



## Conception et production

- ✓ Détail des fonctionnalités
- Workflow de l'application
- Charte graphique
- Les routes
- MCD / MLD / Dico des données / MPD
- Wireframes

## Charte graphique

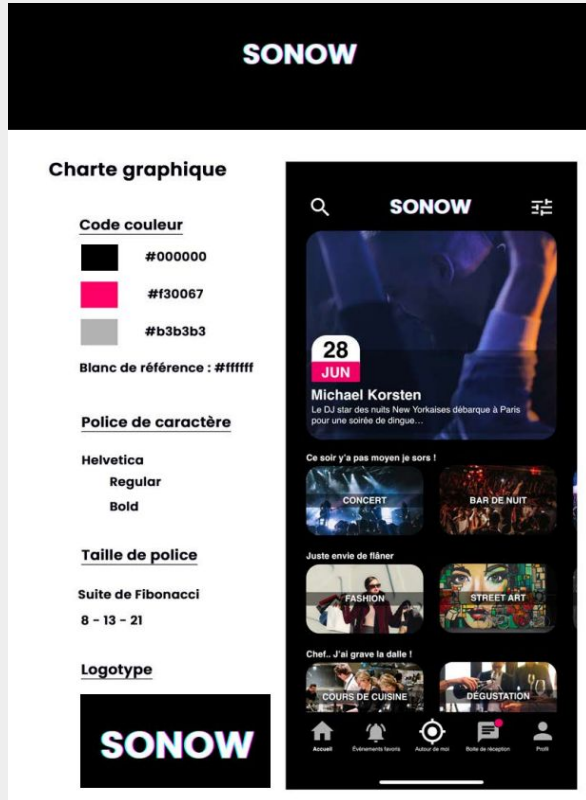


## Conception et production

- ✓ Détail des fonctionnalités
- ✓ Workflow de l'application
- Charte graphique
- Les routes
- MCD / MLD / Dico des données / MPD
- Wireframes

SONOW

## Charte graphique



## Conception et production

- ✓ Détail des fonctionnalités
- ✓ Workflow de l'application
- ✓ Charte graphique
- Les routes
- MCD / MLD / Dico des données / MPD
- Wireframes

SONOW

# Conception et production



## Les routes

URL	HTTP METHODS				CONTROLLERS	METHODS	CONTENTS
	DELETE	POST	PATCH	GET			
/user	✗	✗	✗	✓	user	getAllUsers	Récupérer tous les utilisateurs
/user/:user_id	✓	✓	✓	✓	user	getUserById, updateUser, deleteUser	Récupérer, modifier, supprimer un utilisateur par son ID
/login	✗	✓	✗	✗	user	loginUser	Se connecter à son compte utilisateur
/logout	✗	✗	✗	✓	user	logoutUser	Se déconnecter de son compte utilisateur
/signup	✗	✓	✗	✗	user	createUser	S'inscrire sur l'application en créant un compte utilisateur
/user/search	✗	✗	✗	✓	user	getUserByNickname	Rechercher un autre utilisateur par son surnom
/user/follow	✓	✓	✗	✗	user	followUser, unfollowUser	s'abonner / se désabonner à un utilisateur
/user/:user_id/followers	✗	✗	✗	✓	user	getFollowers	Récupérer la liste des abonnés d'un utilisateur par son ID
/user/:user_id/followed	✗	✗	✗	✓	user	getFollowed	Récupérer la liste des abonnements d'un utilisateur à d'autres comptes par son ID
/event	✗	✓	✗	✓	event	getAllEvents, createEvent	Récupérer la liste des événements
/event/:event_id	✓	✗	✓	✓	event	getOneEventById, updateEvent, deleteEvent	Récupérer, modifier, supprimer un événement par son ID
/event/tag/:tag_id	✗	✗	✗	✓	event	getByTagId	Récupérer les événements d'une catégorie
/event/search	✗	✓	✗	✗	event	getEventsByTitle	Rechercher un événement par son Titre
/event/getbookmarks	✗	✓	✗	✗	event	getEventsByPinUser	Récupérer la liste des événements favoris d'un utilisateur
/event/getattend	✗	✓	✗	✗	event	getEventsByAttendUser	Récupérer la liste des événements auxquels un utilisateur participe
/event/bookmarks	✓	✓	✗	✗	event	addToBookmarks, delToBookmarks	Ajouter / Supprimer un événement en favoris
/event/attend	✓	✓	✗	✗	event	addAttendEvent, delAttendEvent	Participer / Ne plus participer à un événement
/event/:event_slug	✗	✗	✗	✓	event	getOneEventBySlug	Récupérer un événement par son slug
/tag/:tag_id	✗	✗	✗	✓	tag	getOneTag	Récupérer un Tag par son ID
/tag	✗	✗	✗	✓	tag	getAllTags	Récupérer tous les tags
/withevents	✗	✗	✗	✓	tag	getAllTagWithEvents	Récupérer tous les tag avec les événements associés

CRUD : Create = POST // Read = GET // Update = PATCH // Delete = DELETE

Légende : ✓ Route prévue ✗ Route non prévue

SONOW



# Conception et production



## Les routes

URL	HTTP METHODS				CONTROLLERS	METHODS	CONTENTS
	DELETE	POST	PATCH	GET			
/user	✗	✗	✗	✓	user	getAllUsers	Récupérer tous les utilisateurs
/user/:user_id	✓	✓	✓	✓	user	getUserById, updateUser, deleteUser	Récupérer, modifier, supprimer un utilisateur par son ID
/login	✗	✓	✗	✗	user	loginUser	Se connecter à son compte utilisateur
/logout	✗	✗	✗	✓	user	logoutUser	Se déconnecter de son compte utilisateur
/signup	✗	✓	✗	✗	user	createUser	S'inscrire sur l'application en créant un compte utilisateur
/user/search	✗	✗	✗	✓	user	getUserByNickname	Rechercher un autre utilisateur par son surnom
/user/follow	✓	✓	✗	✗	user	followUser, unfollowUser	s'abonner / se désabonner à un utilisateur
/user/:user_id/followers	✗	✗	✗	✓	user	getFollowers	Récupérer la liste des abonnés d'un utilisateur par son ID
/user/:user_id/followed	✗	✗	✗	✓	user	getFollowed	Récupérer la liste des abonnements d'un utilisateur à d'autres comptes par son ID
/event	✗	✓	✗	✓	event	getAllEvents, createEvent	Récupérer la liste des événements
/event/:event_id	✓	✗	✓	✓	event	getOneEventById, updateEvent, deleteEvent	Récupérer, modifier, supprimer un événement par son ID
/event/tag/:tag_id	✗	✗	✗	✓	event	getByTagId	Récupérer les événements d'une catégorie
/event/search	✗	✓	✗	✗	event	getEventsByTitle	Rechercher un événement par son Titre
/event/getbookmarks	✗	✓	✗	✗	event	getEventsByPinUser	Récupérer la liste des événements favoris d'un utilisateur
/event/getattend	✗	✓	✗	✗	event	getEventsByAttendUser	Récupérer la liste des événements auxquels un utilisateur participe
/event/bookmarks	✓	✓	✗	✗	event	addToBookmarks, delToBookmarks	Ajouter / Supprimer un événement en favoris
/event/attend	✓	✓	✗	✗	event	addAttendEvent, delAttendEvent	Participer / Ne plus participer à un événement
/event/:event_slug	✗	✗	✗	✓	event	getOneEventBySlug	Récupérer un événement par son slug
/tag/:tag_id	✗	✗	✗	✓	tag	getOneTag	Récupérer un Tag par son ID
/tag	✗	✗	✗	✓	tag	getAllTags	Récupérer tous les tags
/withevents	✗	✗	✗	✓	tag	getAllTagWithEvents	Récupérer tous les tag avec les événements associés

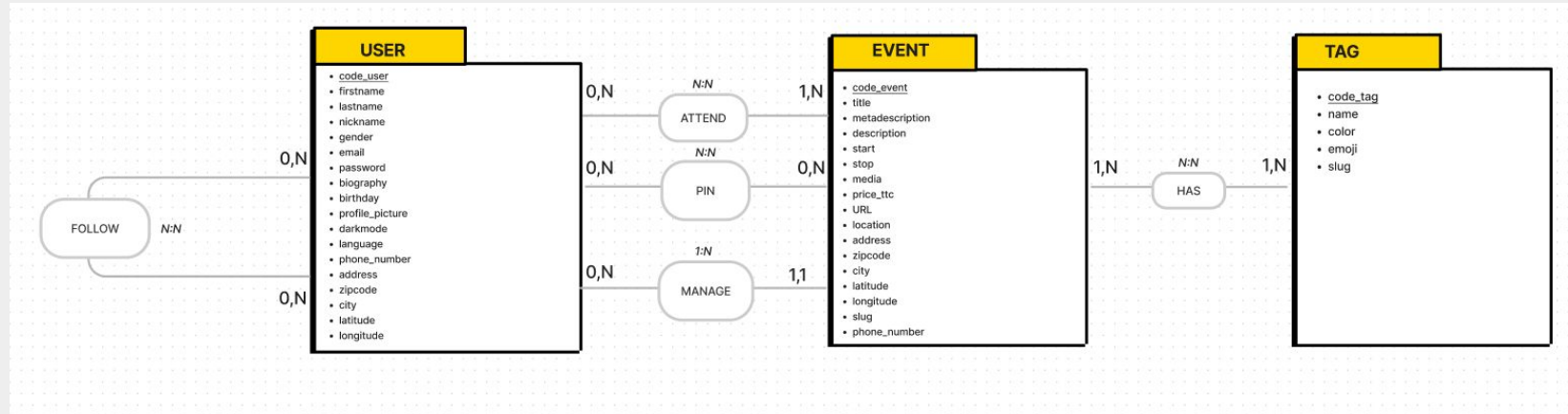
CRUD : Create = POST // Read = GET // Update = PATCH // Delete = DELETE

Légende : ✓ Route prévue ✗ Route non prévue

SONOW

# Conception et production

## ● MCD

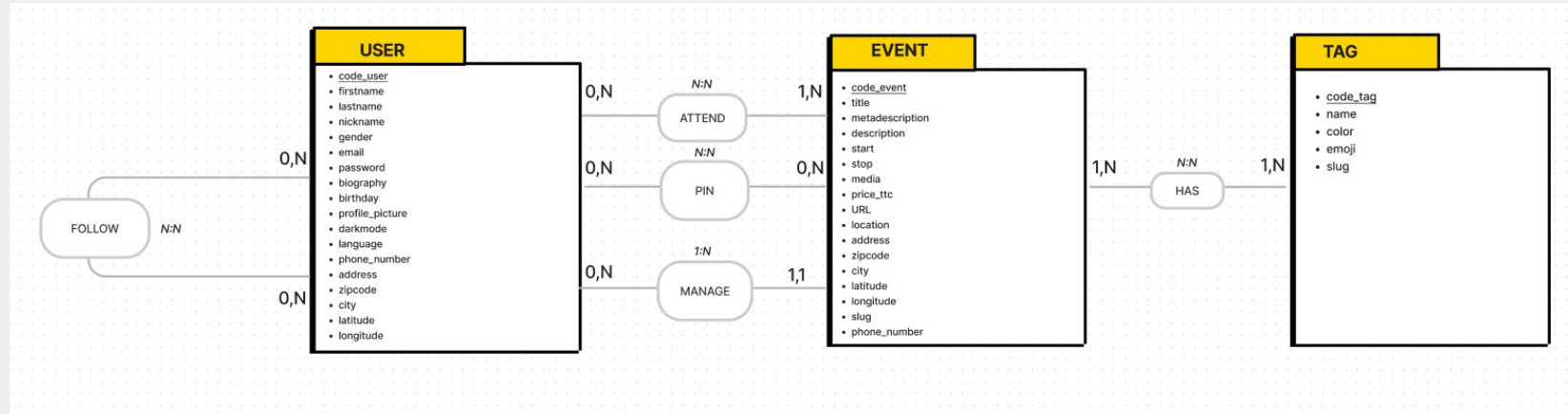


Entités / Propriétés

Relations (verbes)

Cardinalités

# Conception et production



Entités / Propriétés

Relations (verbes)

Cardinalités

# Conception et production

## ● MLD

**USER :** code\_user, firstname, lastname, nickname, gender, email, password, biography, birthday, profile\_picture, darkmode, language, phone\_number, address, zipcode, city, latitude, longitude

**EVENT :** code\_event, title, metadescription, description, start, stop, location, address, zipcode, city, latitude, longitude, slug, phone\_number, media, price\_ttc, URL, #code\_user\_manager

**TAG :** code\_tag, name, color, emoji, slug

**USER\_PIN\_EVENT :** #code\_user, #code\_event

**USER\_ATTEND\_EVENT :** #code\_user, #code\_event

**EVENT\_HAS\_TAG :** #code\_event, #code\_tag

**USER\_FOLLOW\_USER :** #code\_user, #code\_user

clé primaire

#clé étrangère



# Conception et production

## ● MLD

**USER :** code\_user, firstname, lastname, nickname, gender, email, password, biography, birthday, profile\_picture, darkmode, language, phone\_number, address, zipcode, city, latitude, longitude

**EVENT :** code\_event, title, metadescription, description, start, stop, location, address, zipcode, city, latitude, longitude, slug, phone\_number, media, price\_ttc, URL, #code\_user\_manager

**TAG :** code\_tag, name, color, emoji, slug

**USER\_PIN\_EVENT :** #code\_user, #code\_event

**USER\_ATTEND\_EVENT :** #code\_user, #code\_event

**EVENT\_HAS\_TAG :** #code\_event, #code\_tag

**USER\_FOLLOW\_USER :** #code\_user, #code\_user

clé primaire

#clé étrangère

Table de liaison



# Conception et production



MLD

**USER :** code\_user, firstname, lastname, nickname, gender, email, password, biography, birthday, profile\_picture, darkmode, language, phone\_number, address, zipcode, city, latitude, longitude

**EVENT :** code\_event, title, metadescription, description, start, stop, location, address, zipcode, city, latitude, longitude, slug, phone\_number, media, price\_ttc, URL, #code\_user\_manager

**TAG :** code\_tag, name, color, emoji, slug

**USER\_PIN\_EVENT :** #code\_user, #code\_event

**USER\_ATTEND\_EVENT :** #code\_user, #code\_event

**EVENT\_HAS\_TAG :** #code\_event, #code\_tag

**USER\_FOLLOW\_USER :** #code\_user, #code\_user

clé primaire

#clé étrangère

Table de liaison

# Conception et production

## ● Dictionnaire des données

**TABLE USER**

CHAMPS	TYPE	SPECIFICITES	DESCRIPTIONS
id	INTEGER	GENERATED ALWAYS PRIMARY KEY	determining user
firstname	TEXT	NOT NULL	first name
lastname	TEXT	NOT NULL	last name
nickname	TEXT	NOT NULL UNIQUE	nickname
gender	TEXT		gender
email	TEXT	NOT NULL UNIQUE	email of the user
password	TEXT	NOT NULL	connection password
biography	TEXT		various informations the user
birthday	DATE		birthdate
default_profile_picture	TEXT	NOT NULL DEFAULT 'default_profile_picture.webp'	profile photo af the user
language	TEXT	NOT NULL DEFAULT 'FR'	language
darkmode	BOOLEAN	NOT NULL DEFAULT true	dark or light mode choice
address	TEXT		adress of the user
zipcode	TEXT		zip code of the user
city	TEXT		city of the user
latitude	FLOAT		determining latitude set for position
longitude	FLOAT		determining longitude set for position
phone_number	TEXT		phone number of the user

# Conception et production

## ● Dictionnaire des données

**TABLE TAG**

CHAMPS	TYPE	SPECIFICITES	DESCRIPTIONS
id	INTEGER	GENERATED ALWAYS PRIMARY KEY	détermining tag
name	TEXT	NOT NULL	tag name
color	TEXT	NOT NULL	tag color
emoji	TEXT	NOT NULL	tag emoji
slug	TEXT	NOT NULL UNIQUE	slug of the tag

**USER\_PIN\_EVENT**

CHAMPS	TYPE	SPECIFICITES	DESCRIPTIONS
code_user	INTEGER	FOREIGN KEY REFERENCES user(id)	ID of the user in his table
code_event	INTEGER	FOREIGN KEY REFERENCES event(id)	Id of the event in his table

# Conception et production

## ✓ Dictionnaire des données

**TABLE TAG**

CHAMPS	TYPE	SPECIFICITES	DESCRIPTIONS
id	INTEGER	GENERATED ALWAYS PRIMARY KEY	détermining tag
name	TEXT	NOT NULL	tag name
color	TEXT	NOT NULL	tag color
emoji	TEXT	NOT NULL	tag emoji
slug	TEXT	NOT NULL UNIQUE	slug of the tag

**USER\_PIN\_EVENT**

CHAMPS	TYPE	SPECIFICITES	DESCRIPTIONS
code_user	INTEGER	FOREIGN KEY REFERENCES user(id)	ID of the user in his table
code_event	INTEGER	FOREIGN KEY REFERENCES event(id)	Id of the event in his table

BEGIN;

DROP TABLE IF EXISTS public.user, public.event, public.tag, public.event\_has\_tag,  
public.user\_pin\_event, public.user\_attend\_event, public.user\_follow\_user;

```
CREATE TABLE IF NOT EXISTS "user"
(
  "id" integer NOT NULL UNIQUE GENERATED ALWAYS AS IDENTITY (
    INCREMENT 1 START 1 MINVALUE 1 CACHE 1 ),
  "firstname" text NOT NULL,
  "lastname" text NOT NULL,
  "nickname" text NOT NULL UNIQUE,
  "gender" text,
  "email" text NOT NULL UNIQUE,
  "password" text NOT NULL,
  "biography" text,
  "birthday" date,
  "profile_picture" text DEFAULT 'default_profile_picture.webp',
  "language" text DEFAULT 'FR',
  "darkmode" boolean NOT NULL DEFAULT true,
  "phone_number" text,
  "address" text,
  "zipcode" text,
  "city" text,
  "latitude" float,
  "longitude" float,
  "created_at" timestampz DEFAULT CURRENT_TIMESTAMP,
  "update_at" timestampz
);
```

# Conception et production

- ✓ Détail des fonctionnalités
- ✓ Workflow de l'application
- ✓ Charte graphique
- ✓ Les routes
- MCD / MLD / Dico des données / MPD
- Wireframes



```
BEGIN;
```

```
DROP TABLE IF EXISTS public.user, public.event, public.tag, public.event_has_tag,  
public.user_pin_event, public.user_attend_event, public.user_follow_user;
```

```
CREATE TABLE IF NOT EXISTS "user"  
(  
  "id" integer NOT NULL UNIQUE GENERATED ALWAYS AS IDENTITY (  
    INCREMENT 1 START 1 MINVALUE 1 CACHE 1 ),  
  "firstname" text NOT NULL,  
  "lastname" text NOT NULL,  
  "nickname" text NOT NULL UNIQUE,  
  "gender" text,  
  "email" text NOT NULL UNIQUE,  
  "password" text NOT NULL,  
  "biography" text,  
  "birthday" date,  
  "profile_picture" text DEFAULT 'default_profile_picture.webp',  
  "language" text DEFAULT 'FR',  
  "darkmode" boolean NOT NULL DEFAULT true,  
  "phone_number" text,  
  "address" text,  
  "zipcode" text,  
  "city" text,  
  "latitude" float,  
  "longitude" float,  
  "created_at" timestampz DEFAULT CURRENT_TIMESTAMP,  
  "update_at" timestampz  
);
```

# Conception et production

- ✓ Détail des fonctionnalités
- ✓ Workflow de l'application
- ✓ Charte graphique
- ✓ Les routes
- MCD / MLD / Dico des données / MPD
- Wireframes

## Wireframe Mobile

Connexion / Mobile



## Conception et production

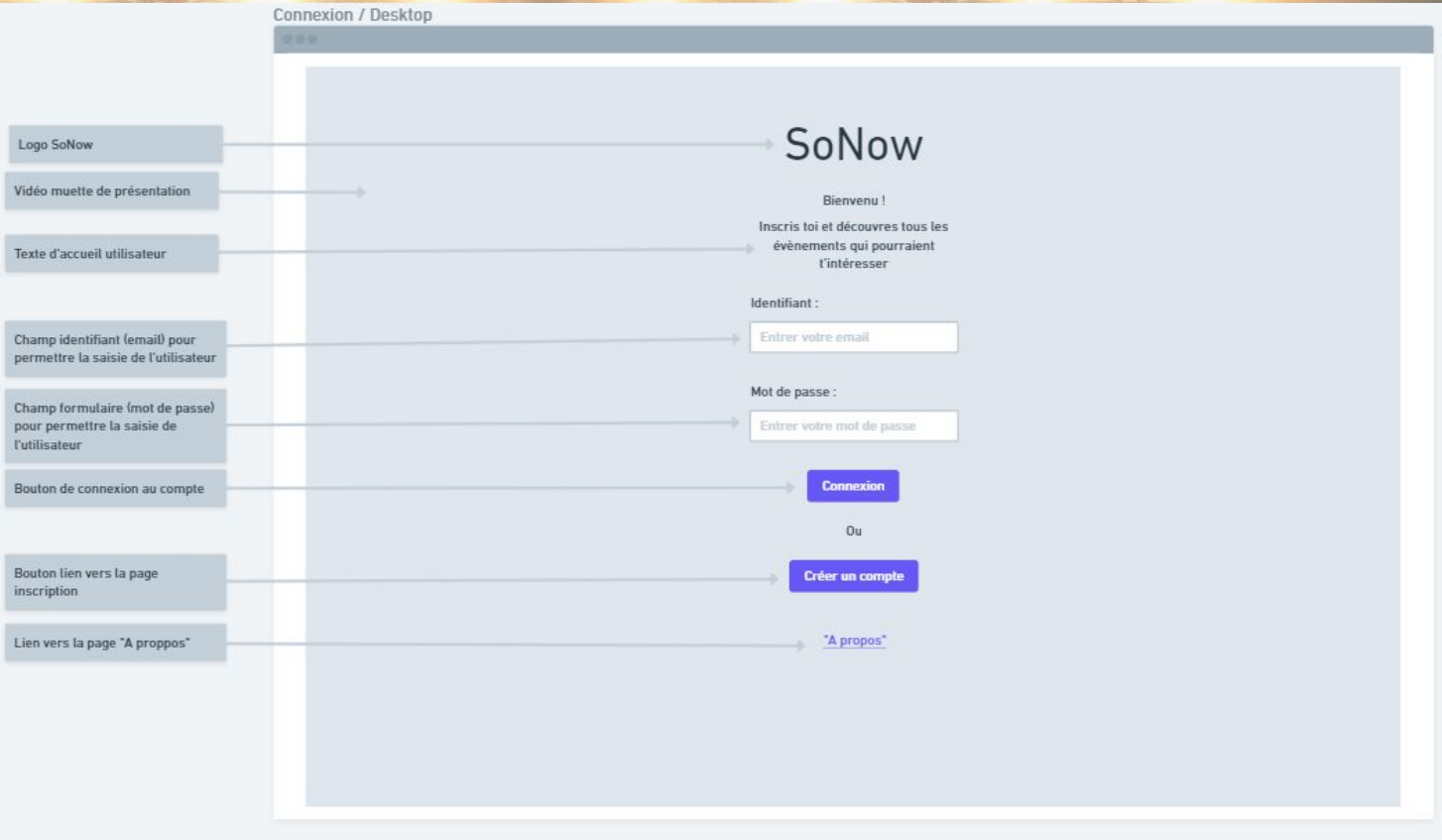
- ✓ Détail des fonctionnalités
- ✓ Workflow de l'application
- ✓ Charte graphique
- ✓ Les routes
- ✓ MCD / MLD / Dico des données / MPD
- Wireframes

SONOW

# Conception et production



## Wireframes Desktop



# Fonctionnalités représentatives

> Se connecter/déconnecter à son compte utilisateur <

- Côté Front-end
- Côté Back-end



# Fonctionnalités représentatives

> Se connecter/déconnecter à son compte utilisateur <

## ● Côté Front-end

Imports librairie + actions

```
import axios from 'axios';
import { getEvents, SUBMIT_LOGIN, submitLoginSuccess, submitLoginError, LOGOUT } from '../actions';

const loginMiddleware = (store) => (next) => (action) => {

  if (action.type === SUBMIT_LOGIN) {

    next(action);

    const state = store.getState();

    let url = 'https://sonow.herokuapp.com/api/user/login/'

    const config = {
      method: 'post',
      url: url + '?nocache=' + new Date().getTime(),
      headers: {
        'content-type': 'application/json; charset=utf-8',
        'Access-Control-Allow-Origin': 'https://sonow.herokuapp.com/api'
      },
      data: {
        email: state.user.login.emailInput,
        password: state.user.login.passwordInput
      }
    }

    axios(config)
      .then((response) => {
        store.dispatch(submitLoginSuccess(response.data.accessToken, response.data.refreshToken, response.data.user));
        localStorage.setItem('accessToken', `${response.data.accessToken}`);
        localStorage.setItem('refreshToken', `${response.data.refreshToken}`);
        localStorage.setItem('id', `${response.data.user.id}`);
        store.dispatch(getEvents());
      })
      .catch(() => {
        store.dispatch(submitLoginError());
      });
  }
}
```

SONOW



# Fonctionnalités représentatives

> Se connecter/déconnecter à son compte utilisateur <

## ● Côté Front-end

Imports librairie + actions

Détails du code de loginMiddleware

Condition

- └ Récupération du state
- └ Définition URL + config
- └ Requête Axios

```
import axios from 'axios';
import { getEvents, SUBMIT_LOGIN, submitLoginSuccess, submitLoginError, LOGOUT } from '../actions';

const loginMiddleware = (store) => (next) => (action) => {

  if (action.type === SUBMIT_LOGIN) {

    next(action);

    const state = store.getState();

    let url = 'https://sonow.herokuapp.com/api/user/login/'

    const config = {
      method: 'post',
      url: url + '?nocache=' + new Date().getTime(),
      headers: {
        'content-type': 'application/json; charset=utf-8',
        'Access-Control-Allow-Origin': 'https://sonow.herokuapp.com/api'
      },
      data: {
        email: state.user.login.emailInput,
        password: state.user.login.passwordInput
      }
    }

    axios(config)
      .then((response) => {
        store.dispatch(submitLoginSuccess(response.data.accessToken, response.data.refreshToken, response.data.user));
        localStorage.setItem('accessToken', `${response.data.accessToken}`);
        localStorage.setItem('refreshToken', `${response.data.refreshToken}`);
        localStorage.setItem('id', `${response.data.user.id}`);
        store.dispatch(getEvents());
      })
      .catch(() => {
        store.dispatch(submitLoginError());
      });
  }
}
```

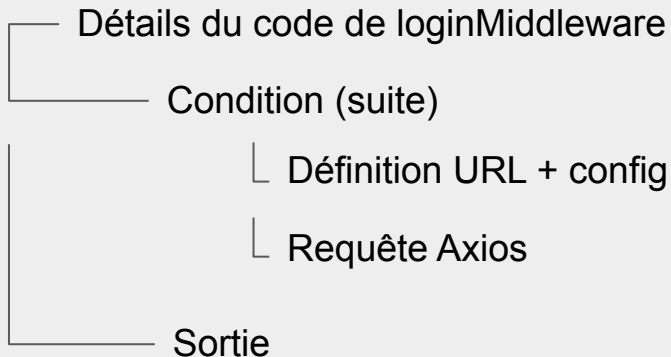
SONOW

# Fonctionnalités représentatives

> Se connecter/déconnecter à son compte utilisateur <

## ● Côté Front-end

Imports librairie + actions



```
} else if (action.type === LOGOUT) {
  next(action);

  let url = 'https://sonow.herokuapp.com/api/user/logout/'

  const config = {
    method: 'get',
    url: url + '?nocache=' + new Date().getTime(),
    headers: {
      'content-type': 'application/json; charset=utf-8',
      'Access-Control-Allow-Origin': 'https://sonow.herokuapp.com/api'
    },
  },
}

axios(config)
  .then(() => {
    localStorage.removeItem('accessToken');
    localStorage.removeItem('refreshToken');
    localStorage.removeItem('id');
  })
  .catch((error) => {
    console.log(error.message);
  });
} else {
  next(action);
}

};

export default loginMiddleware;
```

SONOW

# Fonctionnalités représentatives

> Se connecter/déconnecter à son compte utilisateur <

✔ Côté Front-end

● Côté Back-end

Appel en provenance du Front

Router User

```
const express = require('express');

const validate = require('../validation/validator');
const createSchema = require('../validation/schemas/userCreateSchema');
const updateSchema = require('../validation/schemas/userUpdateSchema');

const router = express.Router();

//Importation du controller des utilisateurs.
const { userController: controller } = require('../controllers');

const controllerHandler = require('../services/controllerHandler');

//Les différentes routes de l'API pour l'utilisateur.

//Route pour récupérer tous les utilisateurs.
router
  .route('/')
  .get(controllerHandler(controller.getAllUsers));

//Routes pour créer ou (dé)connecter l'utilisateur.
router
  .route('/login')
  .post(controllerHandler(controller.loginUser));

router
  .route('/signup')
  .post(validate('body', createSchema), controllerHandler(controller.createUser));

router
  .route('/logout')
  .get(controllerHandler(controller.logoutUser));
```

# Fonctionnalités représentatives

> Se connecter/déconnecter à son compte utilisateur <

✓ Côté Front-end

● Côté Back-end

Appel en provenance du Front

Router User

Contrôleur User

```
require('dotenv').config();
const userDatamapper = require('../models/user');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const SECRET_KEY = process.env.ACCESS_SECRET_KEY;
const REFRESH_SECRET_KEY = process.env.REFRESH_SECRET_KEY;

const { ApiError } = require("../services/errorHandler");

module.exports = {

  //Méthode qui permet à l'utilisateur de se connecter.
  async loginUser(req, res) {
    const { email, password } = req.body;

    let user = await userDatamapper.findByEmailForLogin(email);

    if (user) {
      bcrypt.compare(password, user.password, function (err, result) {
        if (result) {
          req.session.user = user

          const expiresIn = 24 * 60 * 60;
          const accessToken = jwt.sign({ user }, SECRET_KEY, { expiresIn: expiresIn });
          const refreshToken = jwt.sign({ user }, REFRESH_SECRET_KEY, { expiresIn: expiresIn });

          res.header('Authorization', 'Bearer ' + accessToken);
          res.header('RefreshToken', 'Bearer ' + refreshToken);

          delete req.session.user.password;
          return res.status(200).json({ accessToken, refreshToken, user: req.session.user });

        } else {
          return res.status(403).json({ status: 'error', statusCode: 403, message: 'Wrong email or password' });
        }
      });
    }
  },
}
```

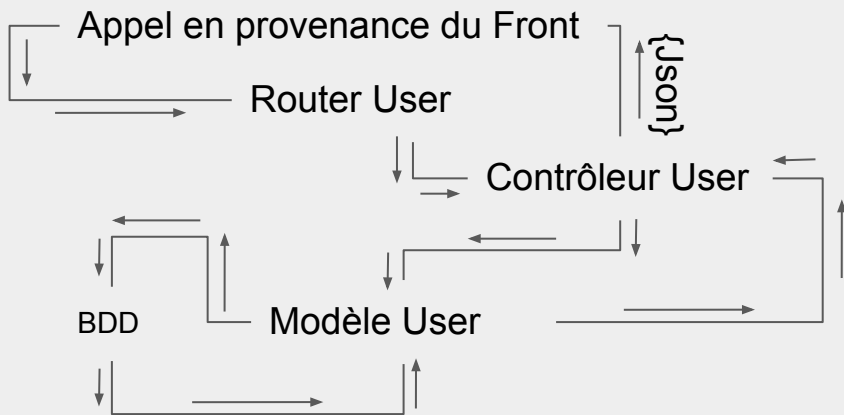
SONOW

# Fonctionnalités représentatives

> Se connecter/déconnecter à son compte utilisateur <

✓ Côté Front-end

● Côté Back-end



```
const client = require("../config/db");

module.exports = {

  async findByEmailForLogin(reqEmail){
    //Je prépare une requête sql séparément pour éviter les injections.
    //J'utilise les jetons sql également par souci de sécurité.
    const preparedQuery = {
      text: `
        SELECT *
        FROM public.user
        WHERE email = $1
      `,
      values: [reqEmail],
    };

    const result = await client.query(preparedQuery);

    if (result.rowCount === 0) {
      return null;
    }

    return result.rows[0];
  },
};
```

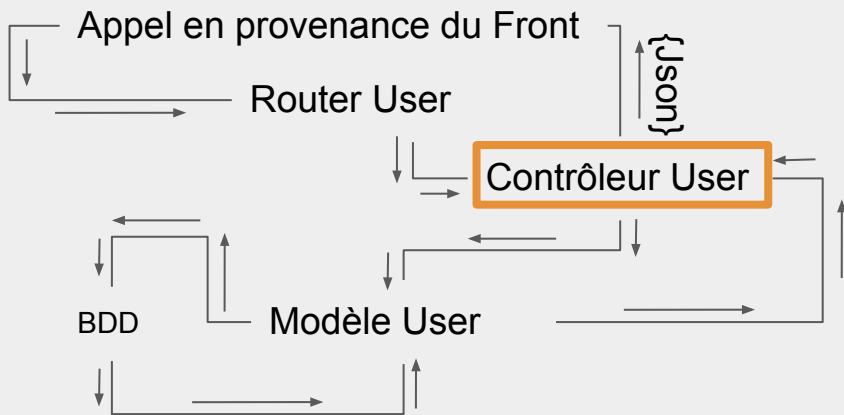


# Fonctionnalités représentatives

> Se connecter/déconnecter à son compte utilisateur <

✓ Côté Front-end

● Côté Back-end



```
const client = require("../config/db");

module.exports = {

  async findByEmailForLogin(reqEmail){
    //Je prépare une requête sql séparément pour éviter les injections.
    //J'utilise les jetons sql également par souci de sécurité.
    const preparedQuery = {
      text: `
        SELECT *
        FROM public.user
        WHERE email = $1
      `,
      values: [reqEmail],
    };

    const result = await client.query(preparedQuery);

    if (result.rowCount === 0) {
      return null;
    }

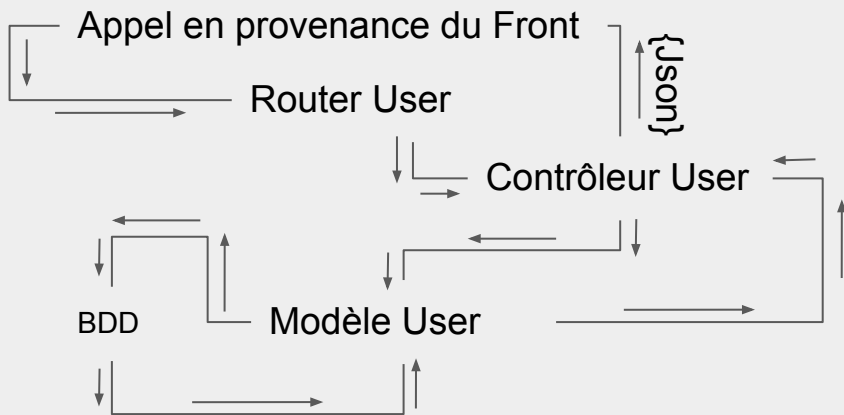
    return result.rows[0];
  },
};
```

# Fonctionnalités représentatives

> Se connecter/déconnecter à son compte utilisateur <

✓ Côté Front-end

✓ Côté Back-end



```
const client = require("../config/db");

module.exports = {

  async findByEmailForLogin(reqEmail){
    //Je prépare une requête sql séparément pour éviter les injections.
    //J'utilise les jetons sql également par souci de sécurité.
    const preparedQuery = {
      text: `
        SELECT *
        FROM public.user
        WHERE email = $1
      `,
      values: [reqEmail],
    };

    const result = await client.query(preparedQuery);

    if (result.rowCount === 0) {
      return null;
    }

    return result.rows[0];
  },
};
```



# Jeu d'essai

Objectif : Tester la fonctionnalité



# Jeu d'essai

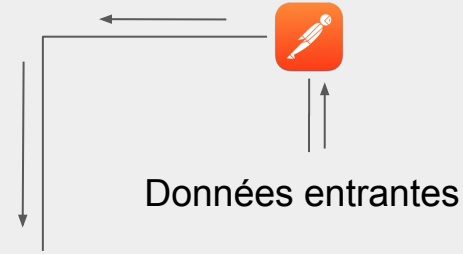
Objectif : Tester la fonctionnalité



Données entrantes

# Jeu d'essai

Objectif : Tester la fonctionnalité



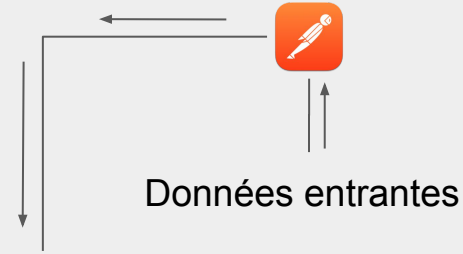
## Données rendues





# Jeu d'essai

Objectif : Tester la fonctionnalité

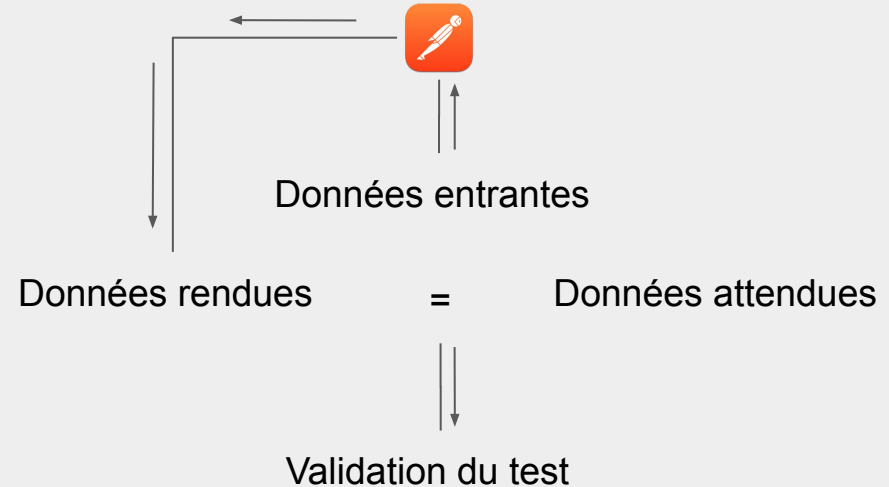


Données rendues = Données attendues



[illegible]

Objectif : Tester la fonctionnalité



# Sécurisation de l'application

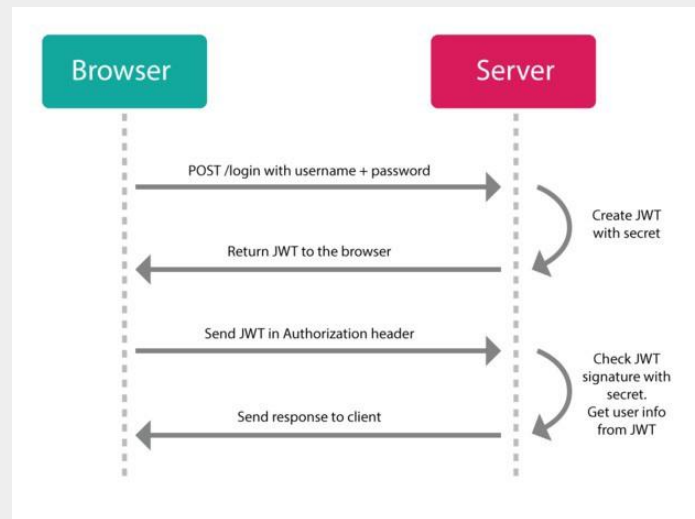
- JWT (Json Web Token)
- CORS (Cross Origin ressources Sharing)
- Requêtes préparées et jeton SQL
- Validateur et Regex pour l'authentification



# Sécurisation de l'application

- JWT (Json Web Token)
- CORS (Cross Origin ressources Sharing)
- Requêtes préparées et jeton SQL
- Validateur et Regex pour l'authentification

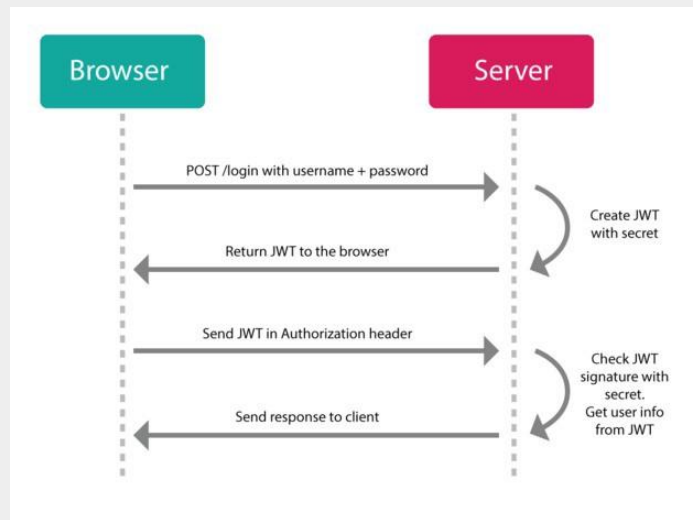
-JWT : Autoriser les transactions ou requêtes entre deux parties.



# Sécurisation de l'application

- ✔ JWT (Json Web Token)
- CORS (Cross Origin ressources Sharing)
- Requêtes préparées et jeton SQL
- Validateur et Regex pour l'authentification

-JWT : Autoriser les transactions ou requêtes entre deux parties.





# Sécurisation de l'application

- ✔ JWT (Json Web Token)
- CORS (Cross Origin ressources Sharing)
- Requêtes préparées et jeton SQL
- Validateur et Regex pour l'authentification

- CORS : Gestion des appels en provenance d'un autre domaine.

Ex: éviter prise de contrôle par client étranger à une API

-Définition au niveau du serveur de l'url autorisée.

# Sécurisation de l'application

- ✔ JWT (Json Web Token)
- ✔ CORS (Cross Origin ressources Sharing)
- Requêtes préparées et jeton SQL
- Validateur et Regex pour l'authentification

- CORS : Gestion des appels en provenance d'un autre domaine.

Ex: éviter prise de contrôle par client étranger à une API

-Définition au niveau du serveur de l'url autorisée.

# Sécurisation de l'application

- ✔ JWT (Json Web Token)
- ✔ CORS (Cross Origin ressources Sharing)
- Requêtes préparées et jeton SQL
- Validateur et Regex pour l'authentification

```
async findByEmailForLogin(reqEmail){
  const preparedQuery = {
    text: `
      SELECT *
      FROM public.user
      WHERE email = $1
    `,
    values: [reqEmail],
  };

  const result = await client.query(preparedQuery);

  if (result.rowCount === 0) {
    return null;
  }

  return result.rows[0];
},
```

# Sécurisation de l'application

- ✔ JWT (Json Web Token)
- ✔ CORS (Cross Origin ressources Sharing)
- ✔ Requêtes préparées et jeton SQL
- Valideur et Regex pour l'authentification

```
async findByEmailForLogin(reqEmail){  
  const preparedQuery = {  
    text: `  
      SELECT *  
      FROM public.user  
      WHERE email = $1  
    `,  
    values: [reqEmail],  
  };  
  
  const result = await client.query(preparedQuery);  
  
  if (result.rowCount === 0) {  
    return null;  
  }  
  
  return result.rows[0];  
},
```

# Sécurisation de l'application

- ✔ JWT (Json Web Token)
- ✔ CORS (Cross Origin ressources Sharing)
- ✔ Requêtes préparées et jeton SQL
- Valideur et Regex pour l'authentification

## Validator

```
const { ApiError } = require("../services/errorHandler");

module.exports = (prop, schema) => async (request, __, next) => {
  try {
    await schema.validateAsync(request[prop]);
    next();
  } catch (error) {
    next(new ApiError(error.details[0].message, { statusCode: 400 }));
  }
};
```



# Sécurisation de l'application

- ✔ JWT (Json Web Token)
- ✔ CORS (Cross Origin ressources Sharing)
- ✔ Requêtes préparées et jeton SQL
- ✔ Valdateur et Regex pour l'authentification

## Validator

```
const { ApiError } = require("../services/errorHandler");

module.exports = (prop, schema) => async (request, __, next) => {
  try {
    await schema.validateAsync(request[prop]);
    next();
  } catch (error) {
    next(new ApiError(error.details[0].message, { statusCode: 400 }));
  }
};
```

## Regex

```
const Joi = require('joi');

module.exports = Joi.object({
  email: Joi.string().required().regex(/^[\w-\.\.]+\.[\w-\.\.]+\.[\w-]{2,4}$/),
  password: Joi.string().required().regex(/^(?=.*[A-Z])(?=.*[a-z])(?=.*\d)(?=.*[!@%$_])([!@%$_\w]{8,15})$/),
  firstname: Joi.string().required(),
  lastname: Joi.string().required(),
  nickname: Joi.string().required(),
}).required();
```

# Sécurisation de l'application

- ✔ JWT (Json Web Token)
- ✔ CORS (Cross Origin ressources Sharing)
- ✔ Requêtes préparées et jeton SQL
- ✔ Validateur et Regex pour l'authentification



SONOW



# Résolution de problème rencontré

Utilisation de sqitch



# Résolution de problème rencontré

Utilisation de sqitch

**bash: ./init.sh: Permission denied**



# Résolution de problème rencontré

Utilisation de sqitch

**bash: ./init.sh: Permission denied**

Recherche avec mots clés

**“Permission denied .sh”**





# Résolution de problème rencontré

Utilisation de sqitch

**bash: ./init.sh: Permission denied**

Recherche avec mots clés

**“Permission denied .sh”**

Réponses sur :

[askubuntu.com](https://askubuntu.com)

[shells.com](https://shells.com)

## Solution

Accorder les droits d'exécution  
à un fichier .sh (sécurité du  
système d'exploitation)

commande à exécuter :  
chmod +x nom\_du\_fichier.sh

## Résolution de problème rencontré

Utilisation de sqitch

**bash: ./init.sh: Permission denied**

Recherche avec mots clés

**“Permission denied .sh”**

Réponses sur :

[askubuntu.com](https://askubuntu.com)  
[shells.com](https://shells.com)

## Solution

Accorder les droits d'exécution à un fichier .sh (sécurité du système d'exploitation)

commande à exécuter :  
`chmod +x nom_du_fichier.sh`

+x exécuter +w écrire +r lire

(g) groupe (u) propriétaire  
(a) tous (o) Autres

## Résolution de problème rencontré

Utilisation de sqitch

**bash: ./init.sh: Permission denied**

Recherche avec mots clés

**“Permission denied .sh”**

Réponses sur :

[askubuntu.com](https://askubuntu.com/shells.com)  
[shells.com](https://shells.com)

# Améliorations

- Sécurité à l'inscription
  - > Envoi d'un mail de confirmation
  - > Gestion Mot de passe oublié
  - > Rôle Admin
  - > Gestion faille XSS (Sanitizer)
- Finalisation raccord routes API au front avant lancement d'améliorations





SONOW

# Conclusion


Synthèse

Accomplissements

Difficultés

Projets





Merci pour votre  
écoute

**SONOW**