Présentation du projet

DRINK GENIUS

Présenté et soutenu par

Stéphane ANDRE

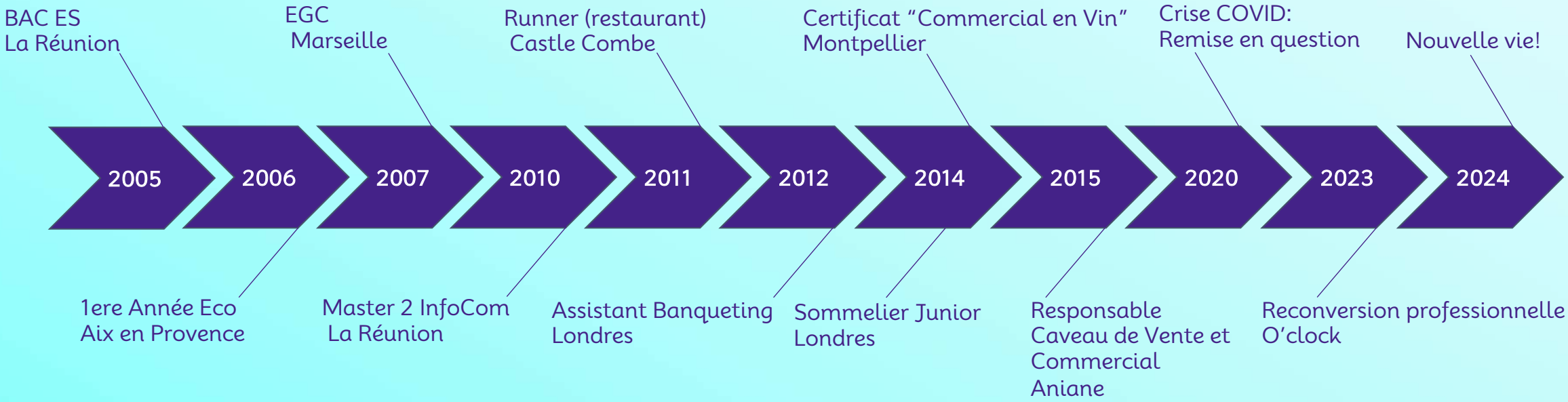
Dans le cadre du Titre Professionnel Développeur Web et Web Mobile

Ecole O'clock – Promotion Yellowstone

Mai 2023 – Octobre 2023

Unforgettable Cocktails

Mon parcours



O'CLOCK

16 semaines : SOCLE

HTML - CSS	JavaScript	Node,Js	Data	Architecture	API	API suite	AdminSys, Déploiement

4 semaines: Spécialisation DATA

API	SQL	NoSQL	Websockets, Datascience

4 semaines: apothéose

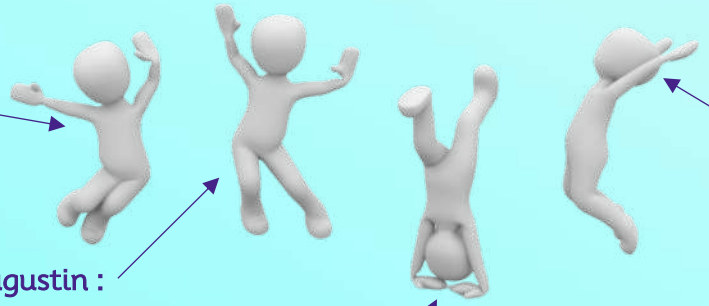
DRINK GENIUS

Le projet : Drink Genius

- Fonctionnalités de l'application:
  - Création de cocktails en mode random
  - Recherche cocktail par filtres

    ➔Initialisation à la mixologie

- The Bar Team:



Mathilde :
- Lead Dev Back
- Referent GIT

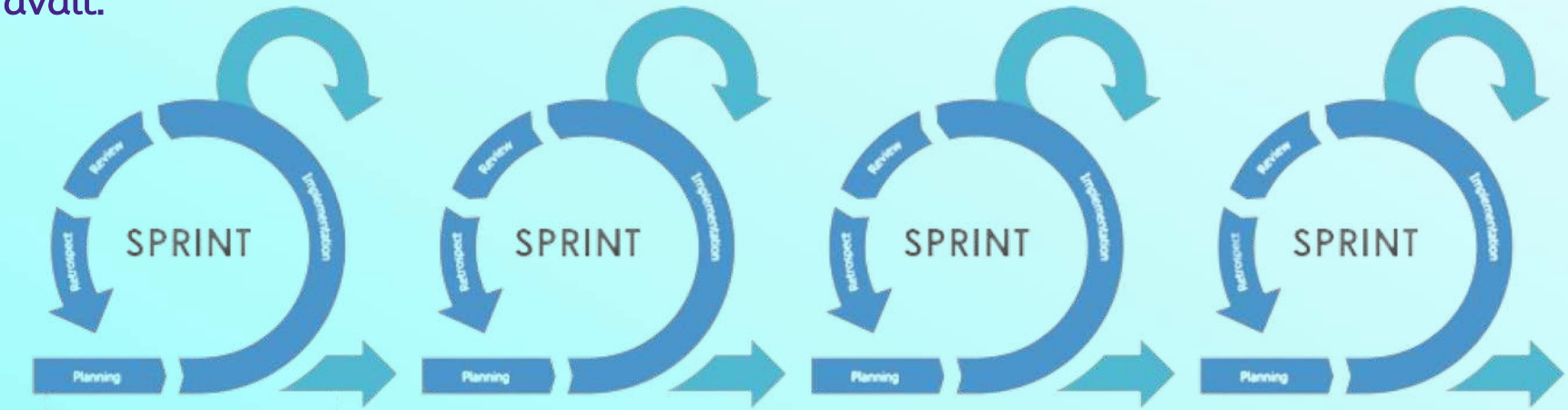
Augustin :
- Lead Dev Front
- Graphiste

Maxime :
- Dev Fullstack
- Co-Scrum Master

Stéphane :
- Dev Fullstack
- Product Owner
- Co-scrum Master

Le projet

- Méthodologie de travail:



- Déroulement:
  - SPRINT 0:
    - Changement de nom
    - Cahier des charges
    - Veilles législatives (alcool sur internet)
  - SPRINT 1:
    - Architecture du projet: MVC
    - Mise en place de la base de données
    - Réalisations des premières fonctionnalités
  - SPRINT 2:
    - Le responsif
    - La sécurité
    - Les modales
  - SPRINT 3:
    - Finir les dernières fonctionnalités
    - Débogage
    - Mise en place des commentaires de code

Le projet

- Les technologies

Communication

Trello

slack

Front end

<%= EJS %>
Embedded JavaScript templating.

JS

Back end

node
JS

Express

JS

Versioning

git

GitHub

Conception

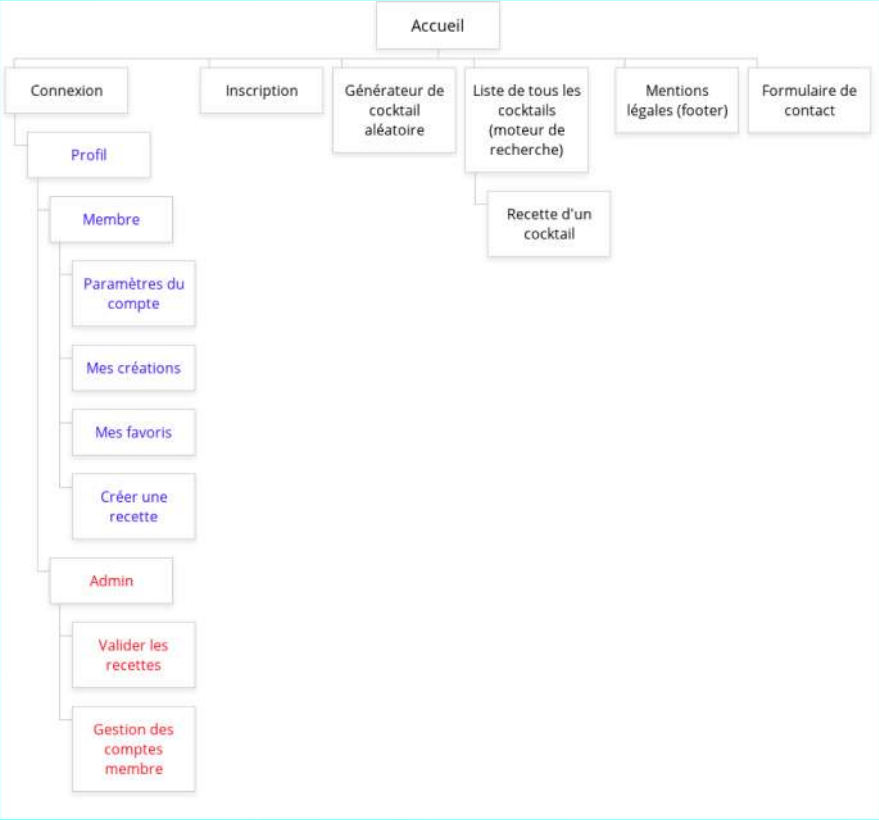
- Le Cahier des Charges:
  - Besoins
  - Objectifs
- Les cibles:
  - « Les aventureux »
  - « Les occasionnels »
- Fonctionnalités essentielles (le Minimum Viable Project)
  - En mode Visiteur
  - En mode Membre
  - En mode Administrateur

- Extrait des users stories

En tant que,,,	Je veux,,,	Afin de,,,
Visiteur – Membre pas encore connecté	pouvoir générer un cocktail avec des ingrédients aléatoires	-
Visiteur – Membre pas encore connecté	trouver une recette de cocktail	-
Visiteur – Membre pas encore connecté	avoir accès au listing des cocktails	Avoir des idées de cocktails
Membre	Pouvoir se connecter	-
Membre	Modifier mon profil	Paramétrer son profil
Membre	Notifier mes cocktails préférés	Les sauvegarder sur mon profil
Admin	Supprimer un compte	Éviter les doublons
Admin	Valider les recettes	Eviter des propos inappropriés
Admin	Proposer des recettes	Enrichir le listing

Conception

- L'arborescence



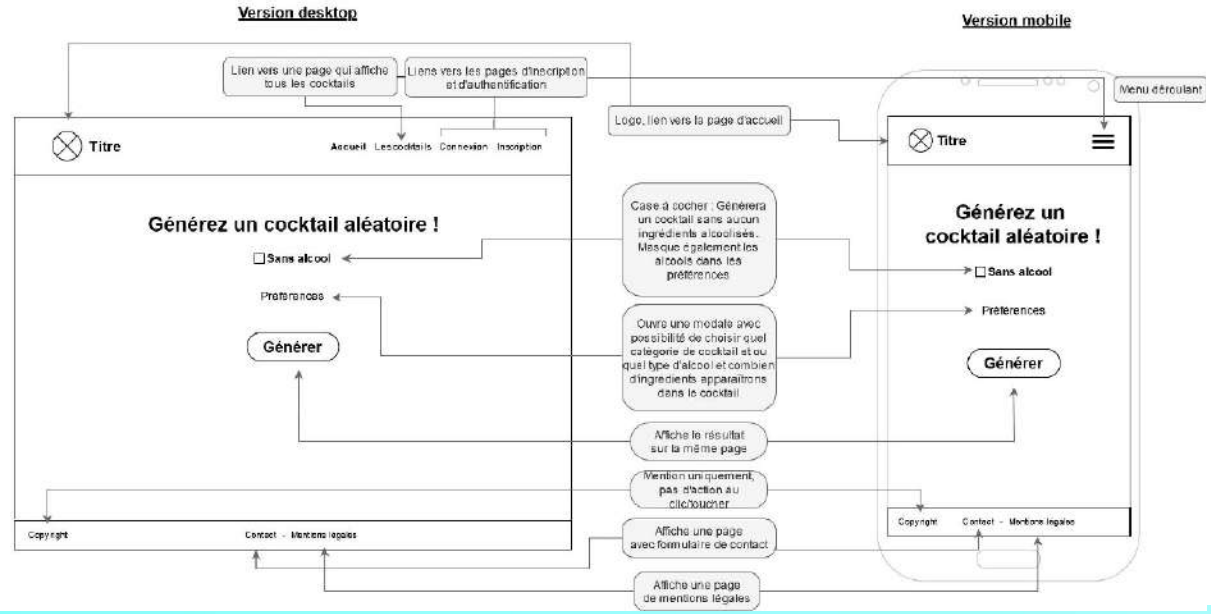
- Extrait des endpoints prévus

Description	Endpoints	Méthode
Visiteur – Membre pas encore connecté		
Récupère les données de la page d'accueil du site	« / »	GET
Envoie les données d'un utilisateur lors de l'inscription	« /signin »	POST
Membre		
Envoi les données d'un membre lors de la connexion	« /login »	POST
Récupère les informations du profil du membre	« /profile/parameters »	GET
Admin		
Envoie les données d'un nouveau cocktail validé par l'admin	« /admin/cocktail »	POST
Supprime toutes les données d'un cocktail	« /admin/cocktail »	DELETE

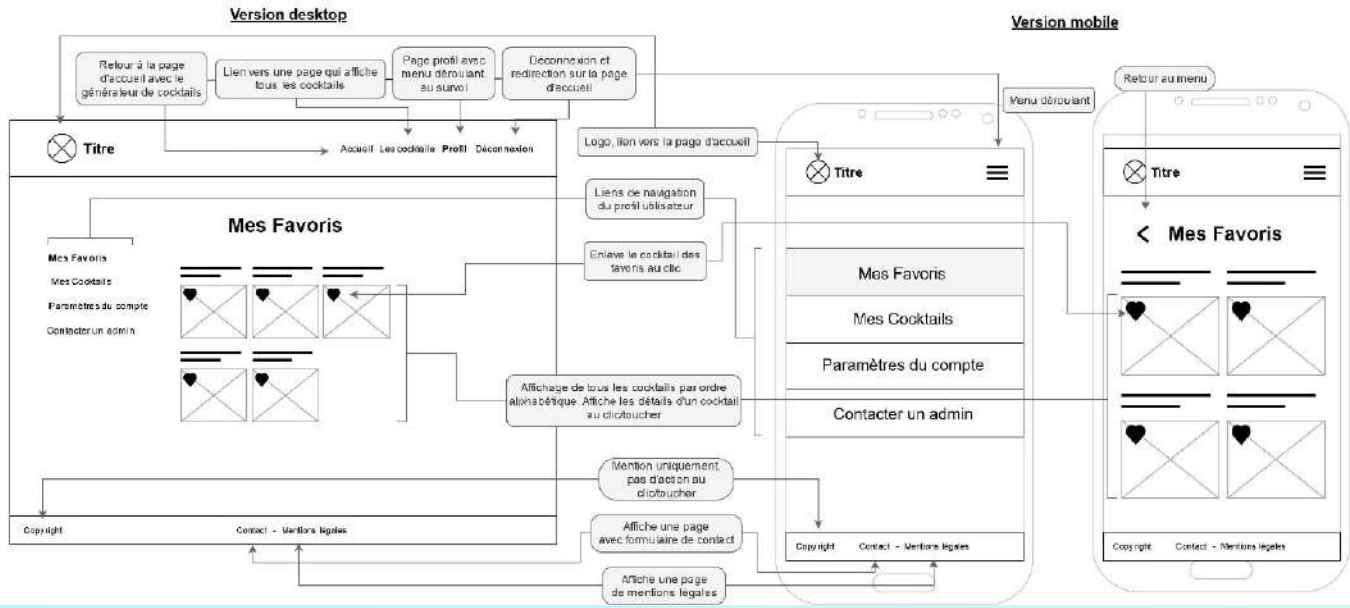
Conception

- Quelques Wireframes

Page d'accueil

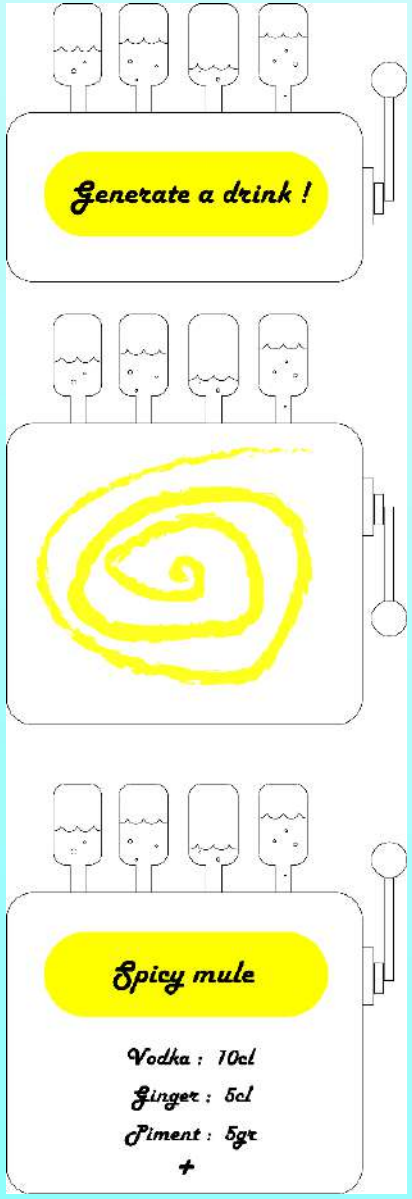


Page profil



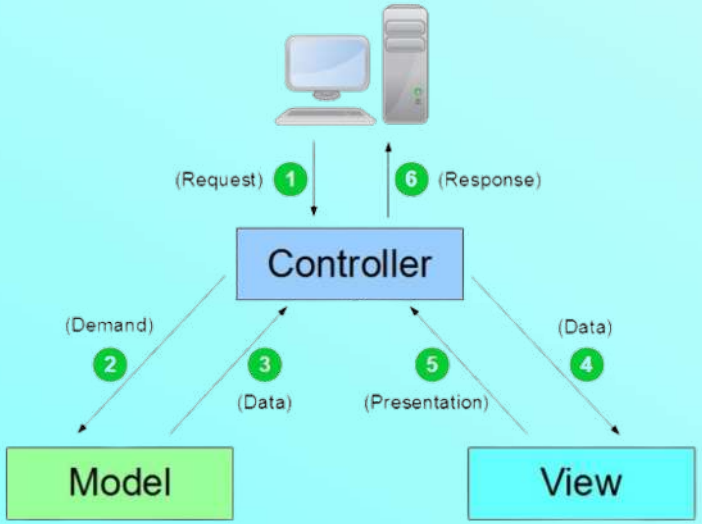
Conception

- Maquette



Conception

- Architecture:

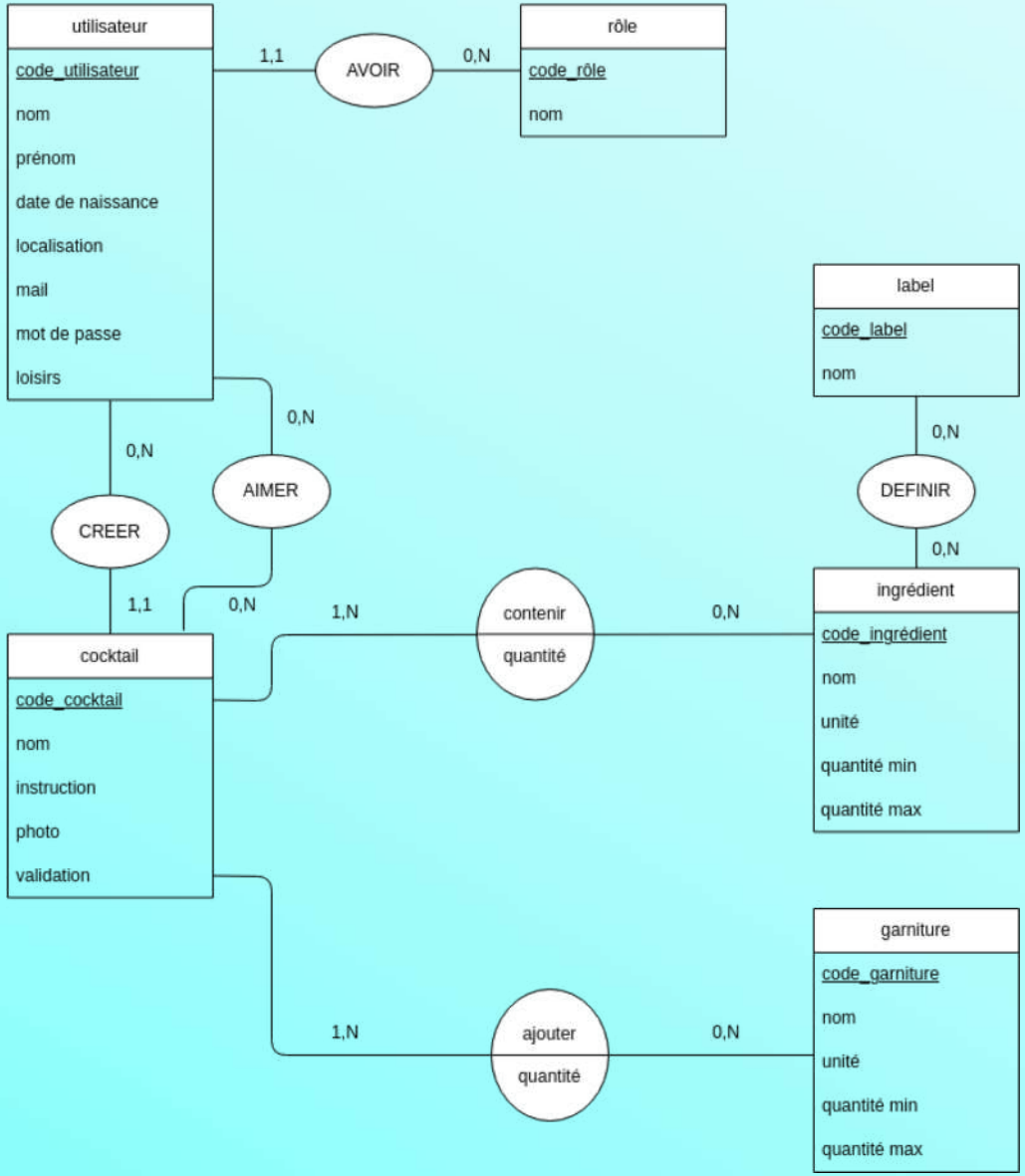


- Exemple:

```
1                                                    2
router.get("/cocktails", cw(cocktailsController.renderAllCocktailsPage));

                                                     3
async renderAllCocktailsPage(req, res){
    const { error, result } = await cocktailDataMapper.getValidatedCocktails();
    const cocktails = result;
    let currentRoute = 'cocktails';
    if(error){
      res.render('erro e', {errorMessa  error});
    } else {                5          6
    res.render('cocktailsListPage', currentRoute});
    }
  },                3
    const cocktailDataMapper = {
    // OBTENIR TOUS LES COCKTAILS VALIDES
    async getValidatedCocktails() {
        let result;
        let error;                                                    4
        try {
            const response = await client.query(`SELECT * FROM cocktail WHERE validation=true`);
            result = response.rows
            if(!result || result.length === 0){
                error = "Aucun cocktail à afficher."
            }
        } catch(err) {
            console.error(err);
            error = "Une erreur s'est produite lors de la récupération des cocktails."
        };
        return {error, result}
    },
    };
```

Conception

- MCD:



Conception

- MLD:
  - User (id, lastname, firstname, birthdate, location, email, password, hobbies, #role_id)
  - Role (id, name)
  - Cocktail (id, name, instruction, picture, validation, #user_id)
  - Contain(#cocktail(id), #ingredient(id), quantity)
  - Add(#cocktail_id, #garnish_id, quantity)
  - Ingredient (id, name, unit, min_quantity, max_quantity)
  - Ingredient_has_label(#ingredient(id), #label(id))
  - User_like_cocktail(#user(id), #cocktail(id))
  - Label (id, name)
  - Garnish (id, name, unit, min_quantity, max_quantity)

- Exemple Dictionnaire des données:

COCKTAIL TABLE			
Field	Type	Specifications	Description
id	INT	GENERATED ALWAYS AS IDENTITY	cocktail's identification
name	TEXT	NOT NULL	cocktail's name
instruction	TEXT	NOT NULL	cocktail's recipe
picture	TEXT		cocktail's picture
validation	BOOLEAN	NOT NULL	cocktail's validation status
user_id	ENTITY	NOT NULL	foreign key linking cocktail to user

Réalisations Personnelles

- FRONT-END: Manipulation du DOM
  - Cas de la SEARCH BAR par nom

```
<!-- le code continue... -->
      <!-- SEARCH BAR -->
      <div class="drink-list-form">

          <input id="searchbar" onkeyup="search_cocktail()"type="text"
          name="search" placeholder="Rechercher par cocktail...">
          <!-- le code continue... -->
          <script src="/js/searchBar.js" defer></script>
```

```
        <!-- COCKTAILS CONTAINER-->
            <div class="cocktails cocktails-container">
                <% cocktails.forEach(cocktail => { %>
                    <a class="cocktail cocktails-container-item" href="/cocktail/<%-cocktail.id%>">
                        <h3 class="cocktail-title"><%-cocktail.name.charAt(0).toUpperCase() +
cocktail.name.slice(1)%></h3>
                        <div class="cocktail-img" style="background-image: url('/images/<%-
cocktail.picture %> ');" alt="Image de <%- cocktail.name %>" id="<%- cocktail.id %>"></div>
                    </a>
                <% }) %>
            </div>
```

```
function search_cocktail() {

    // Récupérer la valeur de la barre de recherche par l'id de la searchbar
  let input = document.getElementById("searchbar").value.toLowerCase();
    // Sélectionnner la classe "cocktail qui englobe le titre et l'img"
  let cocktails = document.getElementsByClassName("cocktail");
    // Boucle for avec variable "i" comme indice, elle commence à 0 jusqu'à la fin de la liste des
cocktails
  for (let i = 0; i < cocktails.length; i++) {
    // Obtenir un élément de la liste
let cocktailTitleElt = cocktails[i].getElementsByClassName("cocktail-title")

//S'il n'y a pas de charactère en commun entre ce qu'on tape et les noms des cocktails
//Il n'affiche rien. Sinon cela affiche le ou les cocktails concernés
    if (!cocktailTitleElt[0].innerHTML.toLowerCase().includes(input)) {
      cocktails[i].style.display = "none";
    } else {
      cocktails[i].style.display = "flex";
    }
  }
}
```

- Récupérer les valeurs
- Sélectionner la classe
- Boucle for
- Obtenir un élément de la liste
- Affichage

Réalisations Personnelles

- BACK-END: requête SQL
  - Filtre par alcool

La route:

```
// Filtre
router.post("/cocktails", cw(cocktailsController.filterCocktailsBySpirits));
```

Le contrôleur:

```
const cocktailDataMapper = require("../models/cocktailDataMapper");
const ingredientDataMapper = require("../models/ingredientDataMapper");

const cocktailsController = {
  // AFFICHE LA PAGE DE TOUS LES COCKTAILS
  async renderAllCocktailsPage(req, res){
    const { error, result } = await cocktailDataMapper.getValidatedCocktails();
    const cocktails = result;
    const spirits = await ingredientDataMapper.getSpiritsName();
    let currentRoute = 'cocktails';
    if(error){
      res.render('errorPage', {errorMessage: error});
    } else {
    res.render('cocktailsListPage', {cocktails, spirits, currentRoute});
    }
  },
  // FILTRE LES COCKTAILS SELON L'ALCOOL
  async filterCocktailsBySpirits(req, res) {
    const spirits_id = (req.body.spirits);
    const cocktailsBySpirit = await cocktailDataMapper.getCocktailBySpirits(spirits_id);
    res.json(cocktailsBySpirit);
},
  // le code continue...
```

Les dataMappeurs (Modèles):

```
const client = require('./dbClient');
const ingredientDataMapper = {
  // OBTENIR LE NOM D'UN INGREDIENT (ici le spiritueux)
    async getSpiritsName(){
        try {
            const result = await client.query(`SELECT ingredient.name AS name, ingredient.id FROM
ingredient WHERE ingredient.id IN (SELECT ingredient_id FROM ingredient_has_label WHERE label_id = 1)`)
            return result.rows;
        } catch (error) {
            return {error: "Erreur s'est produite avec le serveur."}
        }
    },
};
module.exports = ingredientDataMapper;
```

```
async getCocktailBySpirits(ingredient_ids) {
        try {
            const sqlQuery = {
            text: `SELECT DISTINCT ON (cocktail.id) cocktail.*,
cocktail_contain_ingredient.ingredient_id
            FROM cocktail
            LEFT JOIN cocktail_contain_ingredient
            ON cocktail.id = cocktail_contain_ingredient.cocktail_id
            WHERE cocktail_contain_ingredient.ingredient_id =ANY($1)`,
            values: [ingredient_ids]
        }
            const result = await client.query(sqlQuery);
            return result.rows;
        } catch(error) {
            return {error: "Une erreur s'est produite avec le serveur."}
        }
    },
```

La requête SQL : Récupérer le nom du spiritueux

```
SELECT ingredient.name AS name, ingredient.id FROM ingredient WHERE ingredient.id IN (SELECT
ingredient_id FROM ingredient_has_label WHERE label_id = 1) LIMIT 100
```

Search results — Cost: 10ms — Total 23

	name	id
1	rhum blanc	1
2	vodka	7
3	rhum ambré	8
4	gin	14
5	triple sec	19
6	curaçao	21
7	tequila	22
8	amer	25
9	vermouth rouge	26
10	vermouth blanc	27
11	whisky scotch	28

La vue:

Jeux d'essai

- JEUX D'ESSAI 1: SE CRÉER UN COMPTE
- JEUX D'ESSAI 2: AJOUTER DES COCKTAILS EN FAVORIS
- JEUX D'ESSAI 3: CRÉER UN COCKTAIL
- JEUX D'ESSAI 4: EFFACER UN COCKTAIL

C,R,U,D

Exemples de recherche

- Nodemailer => How to set up Nodemailer with Node,Js and a Gmail account?
  - https://miracleio.me/snippets/use-gmail-with-nodemailer/
  - https://stackoverflow.com/questions/71477637/nodemailer-and-gmail-after-may-30-2022/72477193%2372477193

Most times, you would need a pretty quick way to start sending emails in your Node application when you're building for development and that's where Gmail comes in. To get Gmail working with nodemailer, most times, all you have to do is configure your google account by allowing access to "less secure apps" from the security.

With this enabled, you could use your Google email address and password for nodemailer and start sending emails right away.

Unfortunately, this setting is no longer available after May 30th, 2022. This means that when you go back to your account settings this option is no longer available.

Be not dismayed though, thanks to this StackOverflow answer I stumbled on recently, there's a way out. All we need to do now is generate an app password for our google account. We can generate multiple app passwords. These app passwords are usually required when you need to set up and use a third-party email client like Outlook to view emails from your Gmail account.

To generate app passwords, we first have to enable 2-Step Verification on our Google account.

Go to your Google account security settings and enable 2-Step Verification.

Now, you can select the App passwords option to set up a new app password.

In the Select app option, choose Other (custom name) and give it a name of your choice and click on Generate. A 16-digit password will be shown once, you can copy it and keep it somewhere safe. This will be the new password you'll use for nodemailer. Here's the one I generated.

Awesome. Now you can proceed to use your email address and generated password to send emails with nodemailer.

Once Nodemailer is installed, initialize it. Now we create a transporter with gmail and pass our email and generated password in the auth object."

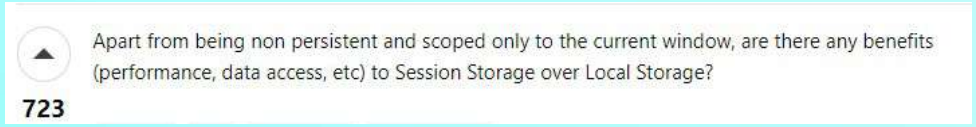
- Modale => How to be sure that it will appear only once when visiting the website the first time?
  - https://stackoverflow.com/questions/62125862/how-to-make-a-component-popup-modal-appear-only-once-using-react-hooks-or-othe
  - https://insidethediv.com/show-popup-modal-only-once-per-session

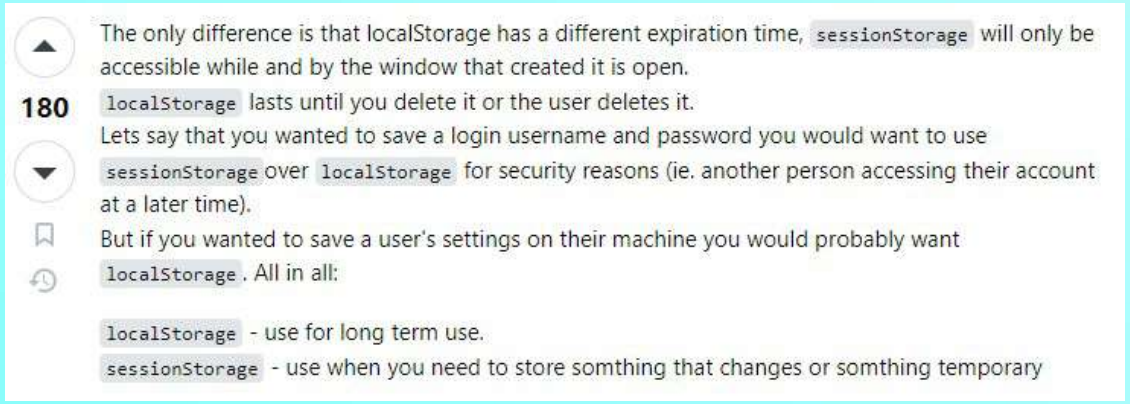
```
export default function Popup() {
        const [visible, setVisible] = React.useState(false);
        useEffect(()=>{
          let pop_status = localStorage.getItem('pop_status');
          if(!pop_status){
            setVisible(true);
            localStorage.setItem('pop_status',1);
          }
        },[])
        if(!visible) return null;

        return (
            <div className={styles.popup} onClick={() => setVisible(false)}>
                {/* <div className={styles.popupInner}> */}
                <div className={styles.popupInner}>
                    <div className={styles.buttonContainer}><Button color="danger" className=
{styles.button}>Okay</Button></div>
                </div>
                {/* </div> */}
            </div>
        )
    }
```

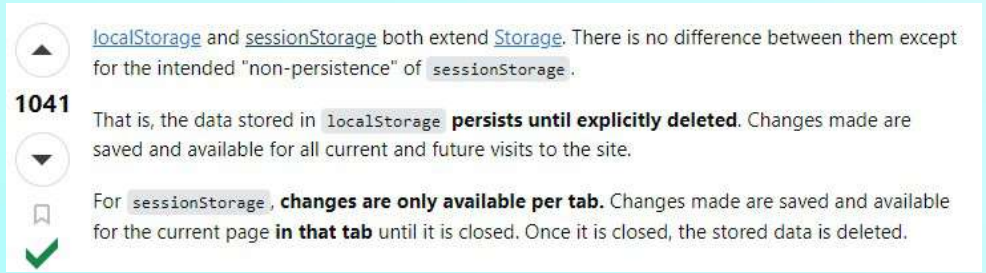
```
setTimeout(showModal,1000);
function showModal(){
  $("#myModal").show()
}
var is_modal_show = sessionStorage.getItem('alreadyShow');
if(is_modal_show != 'alredy shown'){
  $("#myModal").show()
  sessionStorage.setItem('alreadyShow','alredy shown');
}
```

Exemples de recherche

- LocalStorage and SessionStorage,, What are the differences?
  - https://stackoverflow.com/questions/5523140/html5-local-storage-vs-session-storage





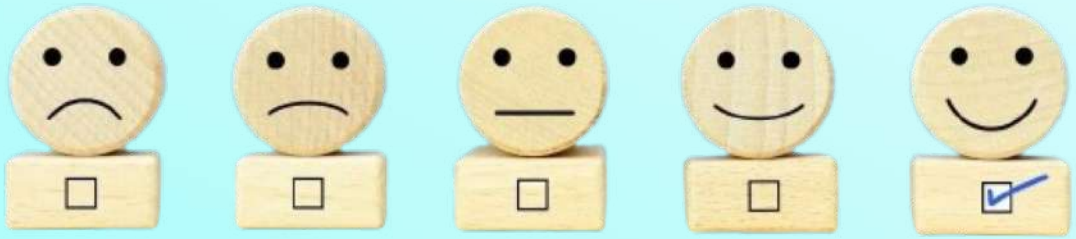


Exemples de recherche

- La sécurité:
  - Ce qu'on a mis en place au niveau du Front:
    - Préparation des inputs
    - Échappement automatique d'EJS
  - Ce qu'on a mis en place au niveau du Back:
    - Bcrypt
    - Joi couplé avec REGEX
    - Express Rate Limite
    - Token JWT
    - Requêtes SQL
    - .env
  - Ce que j'ai appris des recherches après apothéose:
    - Mettre en place un protocole HTTPS avec certificat SSL
    - Mettre en place un CORS Policy (contre les attaques CSRF
    - Mieux exploiter token JWT

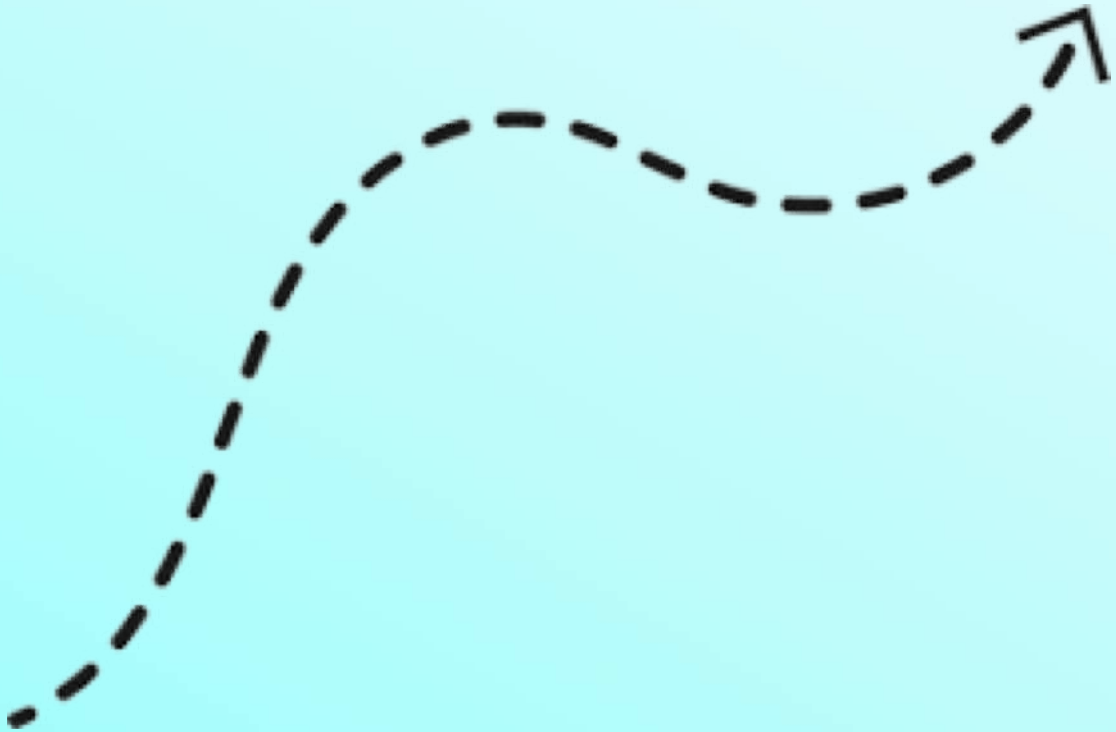
Conclusion

- Bilan projet:
  - Difficultés rencontrées:
    - Organisation
    - FRONT
    - BACK
    - Gestion du temps
  - Les aspects positifs:
    - Le travail fourni
    - Satisfaction personnelle (les « wins »)
    - Le pair-programming



Conclusion

- V2,V3, VX:
  - Améliorer la sécurité
  - Accessibilité
  - Enregistrer un cocktail en mode random
  - Créer un nom de cocktail en mode aléatoire
  - Noter les recettes
  - Pouvoir intégrer des images pour les créations de cocktails
  - Un système de pagination
  - ~~Prévenir l'admin qu'une recette doit être validée~~
  - Refonte du « front » avec un nouveau framework
  - Avoir un système de messagerie entre les membres et avec les administrateurs (websockets?)
  - Pages sur les alcools (un « wikimixologiste »)
  - Des Lives sessions de mixologue
  - Le mode « random » fournit une information supplémentaire,



Conclusion

- Bilan de formation
- Evolution:
  - Chercher du travail? une alternance?
  - Projets personnels

Toutes les images / photos sont libres de droits,

Conclusion

Merci de votre attention