



Dossier de >



Projet(s)

AURÉLIE

DUYNSLAEGER

DWWM

AVRIL 2024

Mem'Téo
& RentUp

mi²
Formation

SOMMAIRE

★ Sommaire	2
★ Introduction	3
★ Frontend : Mem'téo	4
○ Contexte du projet	5
○ Présentation de l'équipe	6
○ Contraintes et livrables attendus	7
○ Environnement technique	9
○ Liste des compétences couverte par le projet	10
■ Wireframes et Maquette de l'application	10
■ Interfaces utilisateurs statiques et dynamiques	12
■ Revue des composants de l'application	16
■	
★ Backend : Rent Up	28
○ Contexte du projet	29
○ Présentation de l'équipe	30
○ Contraintes et livrables attendus	32
○ Environnement humain et technique	33
○ Liste des compétences couverte par le projet	34
■ Création du MCD et de la base de données relationnelle	34
■ Revue des composants d'accès aux données SQL	36
■ Retour sur les composants métier côté serveur	39
■ Déploiement	44
★ Conclusion	47
★ Annexes	48-54

Introduction



Après avoir passé plus d'une dizaine d'années dans le monde de l'hôtellerie-restauration, domaine pour lequel j'ai eu un coup de coeur en sortant de mes études, et où j'ai pu gravir les échelons dans l'une des plus belles maisons de Lille, Meert, j'ai dit au revoir à ce fabuleux corps de métier fin d'année 2022.

Je me suis alors lancée dans un bilan de compétences durant 2 mois, qui m'a amené sur le terrain du numérique. J'ai pu alors intégrer le dispositif Htm'Elles, qui m'a permis de rencontrer des centres de formation, d'entendre des parcours de reconversion, de visiter des écoles et de participer à des ateliers de code. Et c'est ainsi que durant le dernier atelier nous avons pu rencontrer Aymerick, de chez Simplon qui nous a décrit l'Apple Foundation Program. A peine sortie de cet atelier, me voilà à candidater pour faire partie de la prochaine session et être initiée au développement d'applications iOS.

J'ai ainsi pu découvrir les rouages qu'il y avait derrière ce que nous, toutes et tous, utilisons et qui amène simplicité et solutions dans nos quotidiens. Ces 4 semaines riches ont déclenché et affirmé mon envie de reconversion dans le numérique. Ainsi j'ai pu voir ma candidature retenue pour la Nurserie Numérique où j'ai pu découvrir une belle partie des métiers du développement web. Une nouvelle expérience qui a confirmé plus précisément le chemin que je voulais emprunter.

Mon envie était telle que le temps de trouver mon organisme de formation, j'ai suivi chaque jour des cours sur OpenClassRoom afin d'élargir mes premières connaissances et être à même de débiter une année intense en pratique et découvertes.

En septembre 2023, Me voilà ainsi admise chez M2i pour cette réelle mise en route et une très grande soif d'apprendre ces langages et ces façon de concevoir, de penser, de créer, de développer, de se perfectionner, de se remettre en question et surtout de ressentir une belle satisfaction du chemin parcouru et des nouvelles compétences acquises. Étant quelqu'un qui apprécie les challenges et qui possède un état d'esprit apprenant, ces derniers 7 mois ont été un réel plaisir de m'y atteler chaque jour.

J'ai donc choisi de présenter dans ce dossier les 2 projets où j'ai pu travailler en groupe : le premier étant Mem'téo que nous avons développé en 3 semaines avec 3 autres collègues de promotion, et le second Rent Up que j'ai pu développé en stage chez Slamup, en autonomie sous l'oeil avenant de mon tuteur. Les 2 vous démontreront les compétences acquises durant cette formation et pour lesquelles j'ai éprouvé un engouement qui ne cesse de grandir.



Mem'Téo



Contexte du Projet

Mem'téo représente notre projet de fin de formation, une collaboration entre Julie, Angèle, Cyril et moi-même. Ce projet a émergé parmi d'autres idées lors d'un processus de sélection où les équipes ont été formées selon le nombre d'étoiles attribuées à chaque proposition par notre formateur référent. C'est ainsi que Mem'téo a vu le jour !

L'objectif principal de ce projet était de mettre en pratique la méthode Agile et les compétences acquises tout au long de notre formation chez M2i. Notre défi consistait à créer une application météorologique différente, plus ludique et légère à consulter, indépendamment des conditions climatiques. Bien entendu, l'aspect principal demeurerait la précision des données météorologiques, lesquelles sont souvent les plus recherchées. L'enjeu majeur de cette application résidait dans sa capacité à afficher un même en adéquation avec les conditions météorologiques actuelles.

Nous avons découpé notre travail en trois sprints, organisés efficacement sur Trello, ce qui nous a permis de prendre un réel plaisir à concrétiser notre idée en un laps de temps relativement court.



Présentation de l'équipe



Comme je le disais précédemment, nous avons pu travailler à 4 sur ce projet de fin de formation. Les tâches ont été travaillées et distribuées assez naturellement. Ce groupe a été très agréable et riche en échanges, en avancées et en recherche de solutions ! Nous sommes assez fiers de ce que nous avons pu accomplir au cours de ces 3 semaines et appliquer ce que chacun a pu appréhender durant la formation. Le point important aussi est que chacun ait pu être un levier d'amélioration pour l'autre, ce groupe a vraiment **bénéficié d'une belle synergie.**



Contraintes et Livrables



Dans notre groupe, nous avons adopté la méthodologie Agile, favorisant la collaboration et l'implication de tous les membres. Son approche itérative et incrémentale du développement nous a offert à la fois de la flexibilité et de la responsabilité quant aux fonctionnalités proposées et individuellement émises. Nous avons entamé le processus en listant et en priorisant ces fonctionnalités, dans le but de livrer, à la fin du premier sprint, un Minimum Viable Product (MVP).

L'objectif principal était de favoriser la transparence au sein de l'équipe. Chaque matin, nous avons tenu des réunions quotidiennes (Daily), permettant à chacun de savoir où en était le travail des autres membres et d'identifier rapidement les éventuels obstacles. Ces réunions étaient succinctes, se concentrant sur notre progression, nos objectifs pour la journée et les éventuels blocages rencontrés.

En ce qui concerne les livrables, nous avons présenté notre application à la fin des trois sprints, en tentant d'intégrer toutes les fonctionnalités que nous nous étions fixées pour garantir la viabilité du produit. Pour organiser nos sprints et suivre notre progression, nous avons utilisé l'outil Trello, qui s'est avéré être un outil efficace pour la gestion de projet.

Méthodologie

Agile (Scrum) avec 3 sprints de 1 semaine

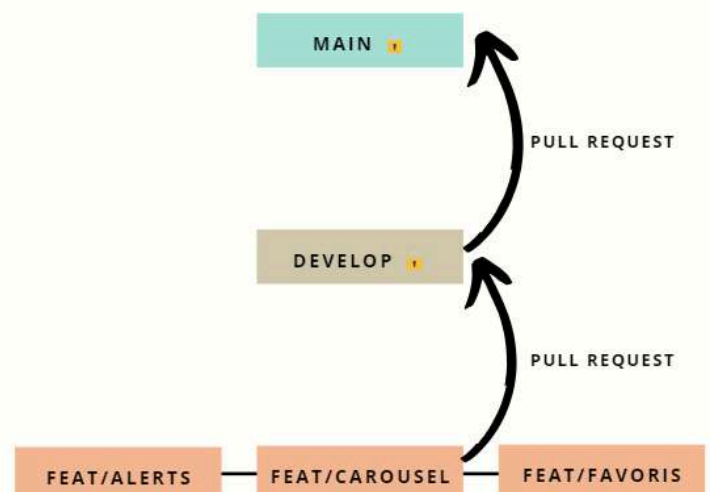
Outils utilisés

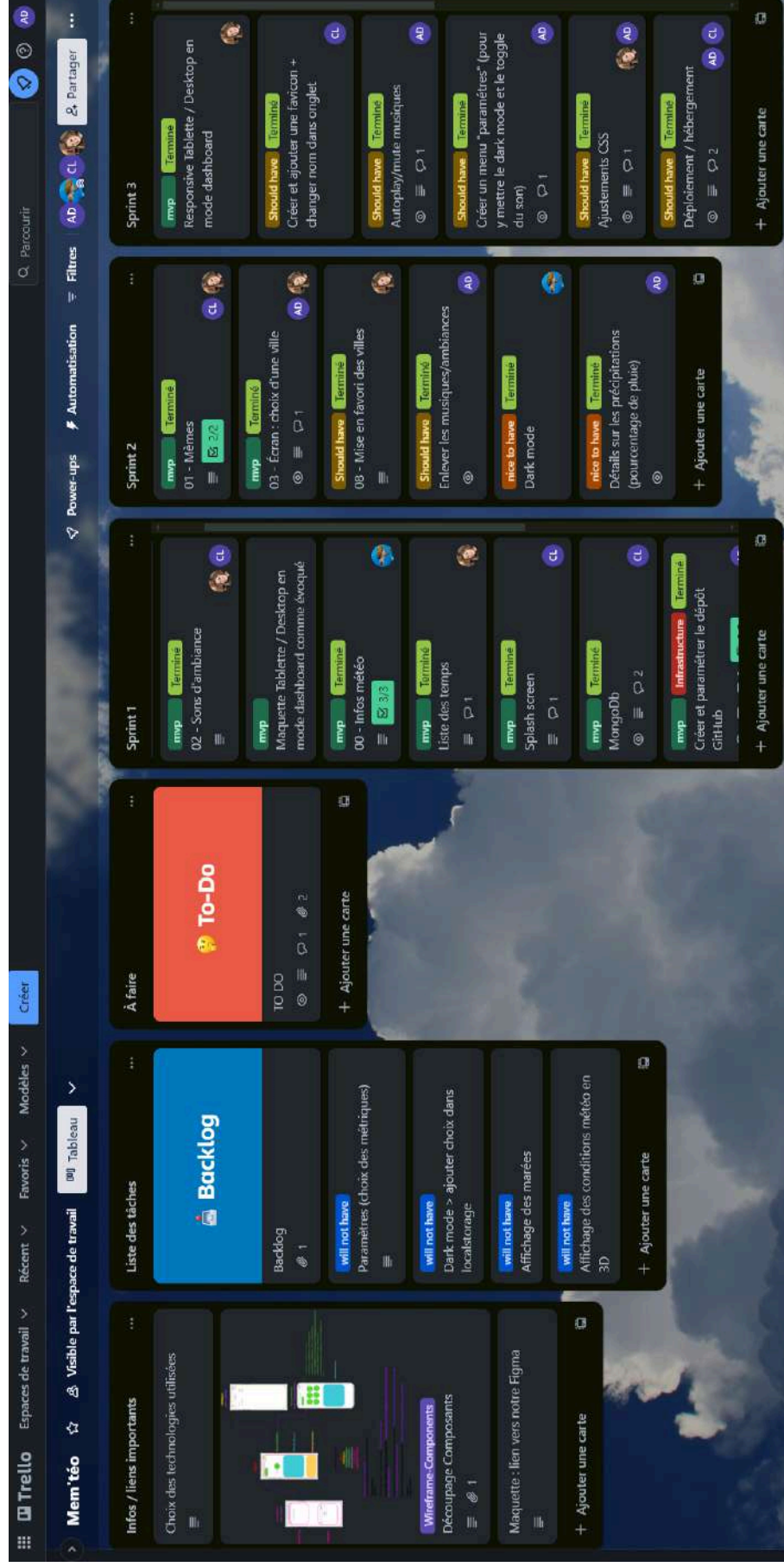
Trello

- Liste des fonctionnalités
- Priorisation (méthode MoSCoW) :
 - Must have (indispensable / vital)
 - Should have (essentiel)
 - Could have (confort)
 - Will not have (luxé)

Git

Teams





Environnement Technique

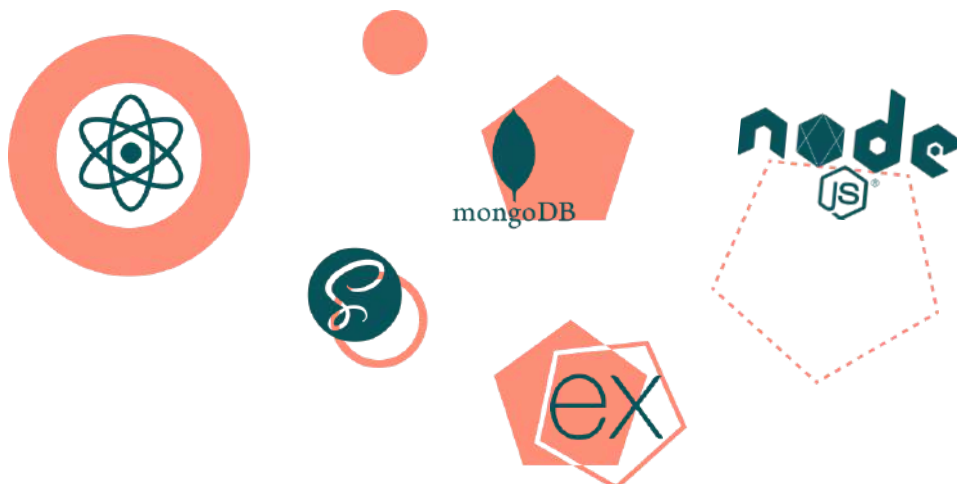


La sphère technique autour de Mem'téo s'est assez vite décidée : en effet, nous partons sur une application qui est consultée la plupart du temps sur mobile. L'idée de base était donc de développer celle-ci sur du React Native.

Sur les conseils de Frédéric qui évoquait sur d'anciens groupes des soucis de compilation avec Android Studio ou Expo, nous ne voulions pas compromettre notre projet sur des choses que nous ne pouvions contrôler ainsi nous en sommes venus à la conclusion de développer celle-ci en ReactJs et, si le temps impartit nous le permettait, de basculer sur le Native.

Concernant les autres stacks : nous avons à l'unanimité voulu travailler le style sans framework Css tels que Tailwind ou Bootstrap, et avons opté pour le scss et utiliser le préprocesseur Sass.

Concernant le côté backend, nous voulions compléter notre stack en créant une Api par le biais de Node Js et Express Js, et en stockant nos données sur MongoDB. Nos données étant les memes et les sons d'ambiance.



Compétences couvertes par le projet

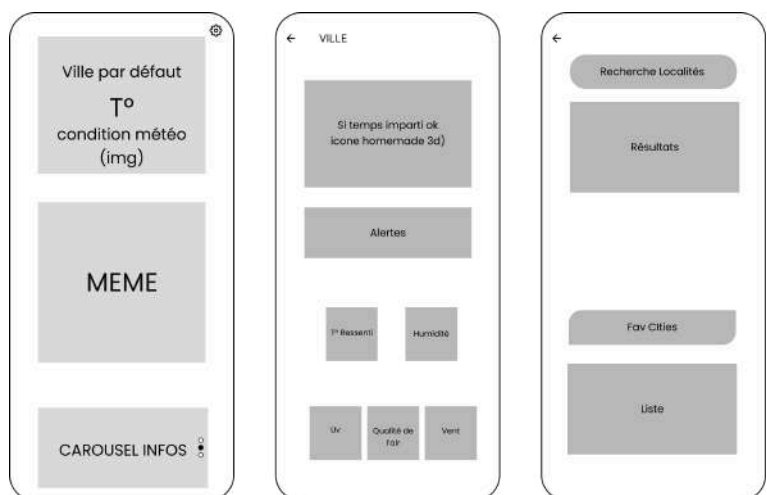
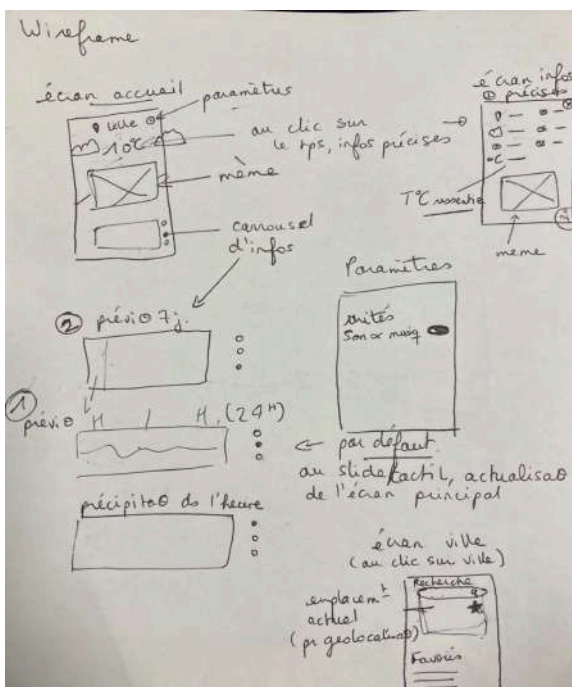
◆ Maquetter une application



Avant de concrétiser notre application, nous avons pris soin de bâtir ses fondations à travers une série d'étapes cruciales.

Nous avons commencé par élaborer au cours de notre premier brainstorming, ce que chacun attend ou attendrait d'une application météo : ce qu'il trouve primordial, ce qu'il n'utilise pas. Se mettre donc à la place de l'utilisateur et esquisser une "user story" afin de définir clairement les fonctionnalités attendues de notre application.

Ensuite, au fur à mesure des échanges, nous avons donné vie à nos idées en créant des wireframes pour représenter la structure de nos pages, principalement en version mobile.

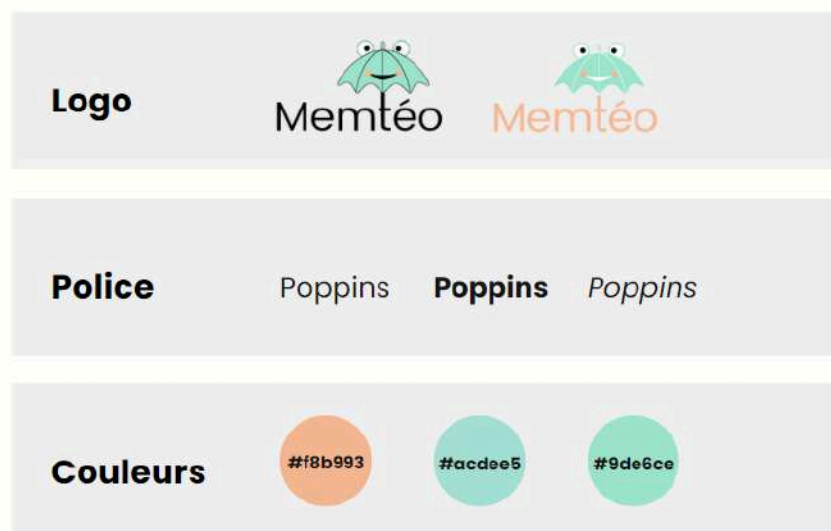


Nous avons aussi complété le benchmark des diverses applications météo qui étaient moins traditionnelles : lecture des commentaires des utilisateurs, retours, engouement.



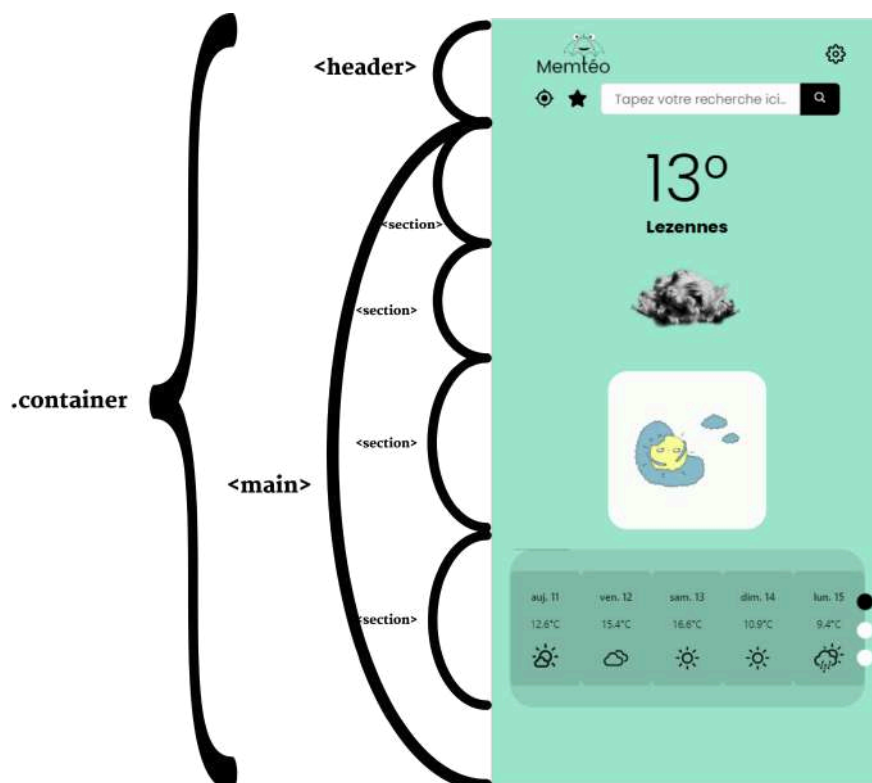
Enfin, nous avons affiné l'aspect visuel de notre application en élaborant une charte graphique et des maquettes, nous permettant ainsi de visualiser concrètement le rendu final, comme en témoignent les captures d'écran de nos wireframes et maquettes.

Charte graphique



Compétences couvertes par le projet

◆ Interfaces Utilisateur Statiques et Dynamiques



Concernant la partie statique de notre application, nous avons procédé à un découpage des maquettes et ainsi pu construire le squelette de celle-ci. Nous avons organisé le contenu en utilisant des balises HTML appropriées telles que `<header>`, `<main>`, `<section>`, pour garantir une hiérarchie logique, appliquer les fondamentaux du développement web et

notre capacité à structurer efficacement une interface utilisateur statique.

```
return (  
  <div className="container">  
    { /* HEADER = logo + searchBox + params (light/dark mode + sound) */ }  
    <header>  
      <Logo isDarkMode={isDarkMode}></Logo>  
  
      { /* Affichage des params */ }  
      <div className="settings"><FiSettings className="settings-icon" onClick={() => showDrawer("right")} /></div>  
      <Drawer title="Paramètres" placement={placement} onClose={onClose} open={open}>...</Drawer>  
  
      { /* composant NavBar qui permet la saisie d'une ville ou la geolocalisation */ }  
      <SearchBox onWeatherInput={handleWeatherInput} setLoadingCity={setLoadingCity} />  
    </header>  
  
    { /* MAIN */ }  
    <main>  
      <div className="group">  
        <section className="currentWeatherForecast">...</section>  
        <section className="currentWeatherImage">...</section>  
        <section className="meme">...</section>  
      </div>  
  
      <div>  
        <section className="carousel">...</section>  
      </div>  
    </main>  
  </div>  
)
```

Cette caractéristique de l'interface utilisateur statique garantit une cohérence totale dans le rendu de la page, indépendamment de l'utilisateur qui y accède. Les éléments tels que la disposition du logo dans le header, l'affichage des données météorologiques, et le positionnement du carrousel restent constants, offrant ainsi une expérience utilisateur uniforme et prévisible pour tous les utilisateurs.

Notre approche de développement a également inclus l'intégration d'éléments visuels essentiels tels que des icônes et des images pour améliorer l'expérience utilisateur. Nous avons utilisé des bibliothèques d'icônes comme React Icons pour garantir une représentation visuelle cohérente. De plus, nous avons veillé à l'accessibilité en intégrant des attributs alt aux images pour une expérience utilisateur inclusive et respectueuse des normes d'accessibilité du web. Cette attention aux détails visuels et fonctionnels a contribué à offrir une interface utilisateur statique attrayante et facile à naviguer pour tous les utilisateurs.

Concernant l'interface utilisateur dynamique, essentiel dans le développement d'applications interactives, réagira et s'adaptera ici donc en temps réel aux actions de l'utilisateur et aux changements de données, offrant ainsi une expérience utilisateur plus engageante et personnalisée, selon leurs besoins et préférences individuels.

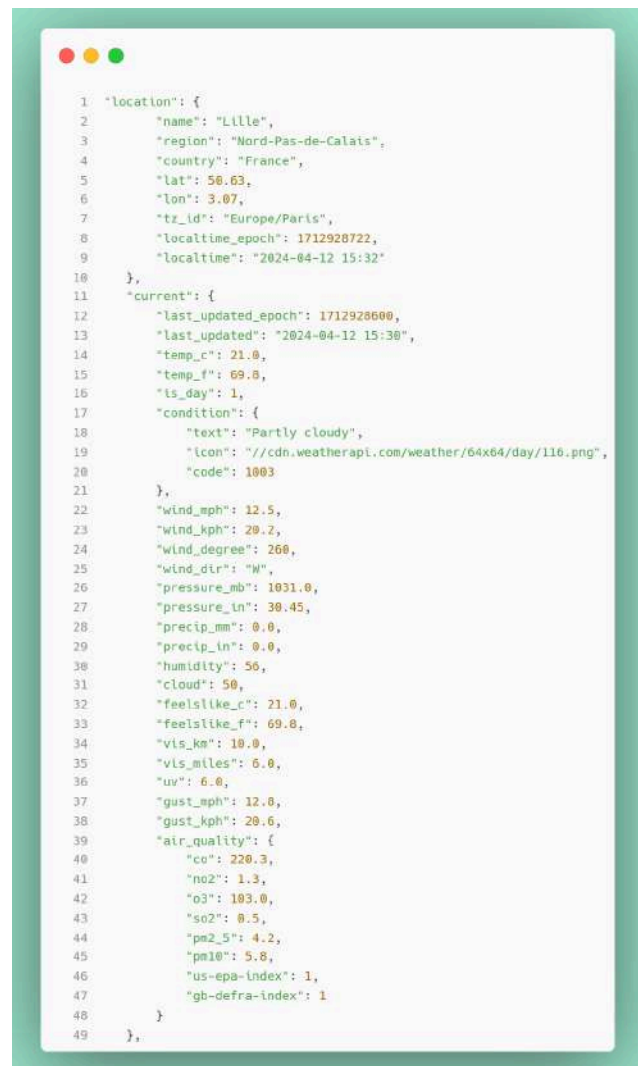
Dans cette partie de la présentation, nous allons explorer comment nous avons intégré des fonctionnalités dynamiques dans notre application météo, en mettant en lumière les avantages qu'elles apportent en termes d'expérience utilisateur et d'engagement .

Les interactions de l'utilisateur et les conditions météorologiques jouent un rôle central dans la dynamique de notre application météo. Par exemple, les actions telles que la recherche de nouvelles localités, la sélection des paramètres de personnalisation et l'exploration des fonctionnalités peuvent modifier instantanément le contenu affiché à l'écran. De plus, les changements dans les conditions météorologiques en temps réel, tels que les variations de température ou les prévisions de pluie, peuvent influencer directement les informations affichées dans notre application, offrant ainsi aux utilisateurs des données météorologiques pertinentes et actualisées.

L'API WeatherApi est au cœur de notre application météo, nous offrant un accès facile à une gamme complète d'informations météorologiques. Avec ses fonctionnalités étendues, telles que la prise en charge des unités de mesure et un plan gratuit complet, elle nous permet de fournir à nos utilisateurs des données précises et actualisées sur les conditions météorologiques.

API qui joue donc un rôle crucial dans la personnalisation de l'expérience utilisateur en fournissant des données météorologiques précises et fiables, quel que soit l'endroit où se trouve l'utilisateur. Je me suis chargée du test de l'Api afin de partager aux collègues ce qu'on en récupérerait, j'ai aussi fait quelques tests d'affichage pour que chacun puisse se faire à l'Objet et ainsi établir une réflexion pour chacune de nos parties.

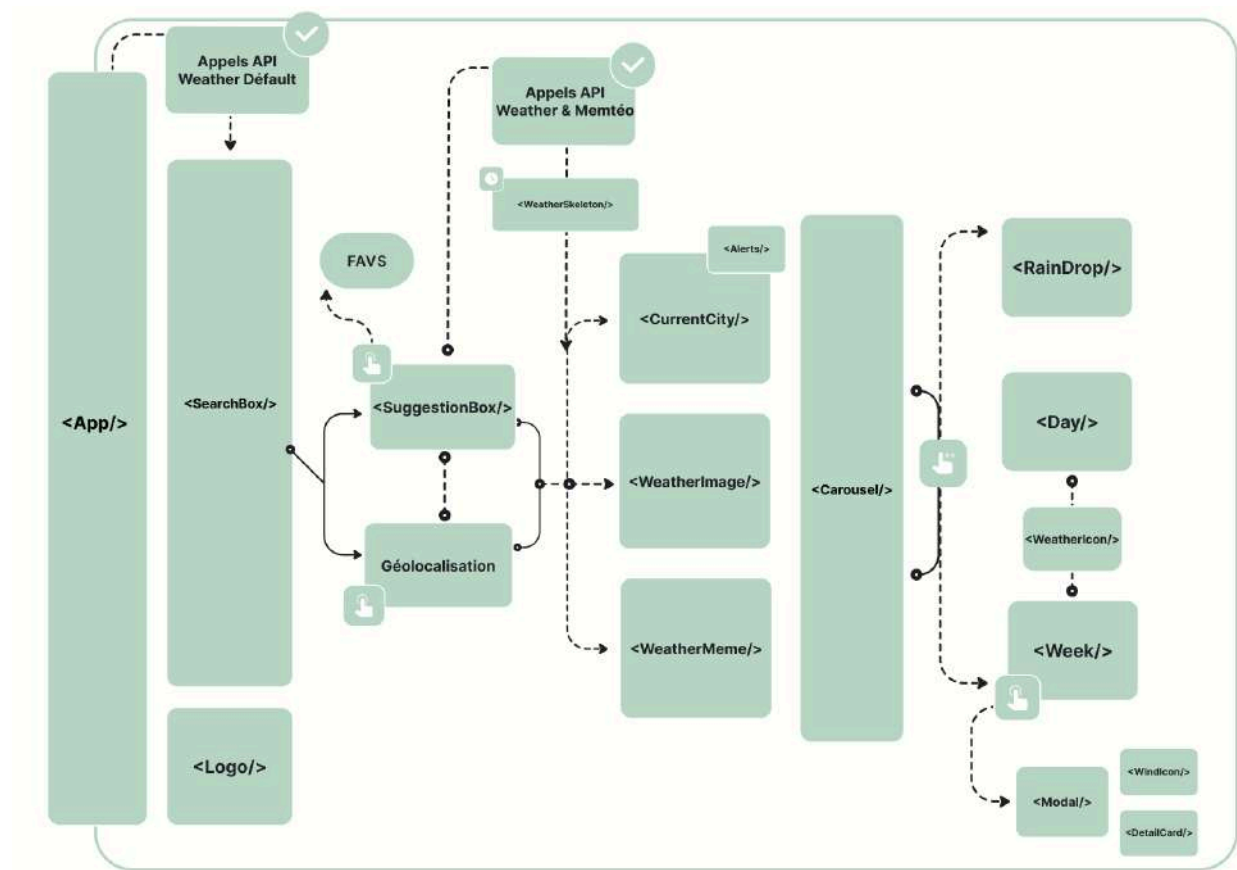
Aussi en se basant sur notre benchmark, nous avons pris la décision de faire un appel par défaut sur la ville de Lille pour améliorer l'accessibilité, la convivialité de notre application dès la première utilisation, et également une expérience universelle pour tous les utilisateurs, quel que soit leur emplacement géographique initial. Car souvent les utilisateurs n'apprécient pas d'avoir une demande de géolocalisation dès leur première visite.



Réponse de l'appel Api.



J'ai donc élaboré un schéma UML détaillé pour comprendre le flux des données à travers notre application météo. Ce schéma représente visuellement les différentes étapes du processus, de la récupération des données de l'API à l'affichage des informations météorologiques dans l'interface utilisateur. Il nous permet de mieux comprendre la logique sous-jacente de notre application et de garantir un flux de données fluide et efficace.



Notre application offre une gamme de paramètres dynamiques, tels que le choix du mode sombre ou clair, vouloir ou non les sons, le dynamisme de la couleur du fond, la géolocalisation, la possibilité de mettre les localités en favoris. Ces paramètres permettent aux utilisateurs de personnaliser leur expérience selon leurs préférences individuelles, améliorant ainsi leur confort d'utilisation et leur satisfaction globale. Nous avons intégré ces options de personnalisation dans notre interface utilisateur de manière intuitive et accessible, permettant ainsi aux utilisateurs de les modifier facilement à tout moment.

Revue des Composants



◆ <SearchBox/>



Passons donc à la revue des principaux composants de notre application.

Nous avons donc vu le premier rendu de App, basé sur l'appel API par défaut. Parlons donc de la première chose que ferait l'utilisateur : rechercher sa localité actuelle ou une localité en particulier, fonctionnalités que renferme donc le SearchBox Component. Deux choix ici s'offrent à lui :

- Géolocalisation → un clic sur l'icône de géolocalisation,
- Rechercher → Input de recherche qui lui permettra à chaque caractère entré, de voir une liste de suggestions, sur laquelle il pourra faire son choix : au clic, sélectionner la ville, ou la marquer en tant que favoris.





Ce composant donc renferme 3 actions :

☀ handleGeolocation

Lorsqu'elle est déclenchée, elle commence par définir `loadingCity` sur `true`, ce qui indique généralement à l'utilisateur qu'une opération est en cours.

Ensuite, elle vérifie si le navigateur prend en charge la géolocalisation à l'aide de `navigator.geolocation`. Si c'est le cas, elle utilise `getCurrentPosition` pour obtenir les coordonnées géographiques actuelles de l'utilisateur. Une fois que les coordonnées sont récupérées

avec succès, la fonction extrait les valeurs de latitude et de longitude à partir de l'objet `position.coords`. Ces coordonnées sont ensuite utilisées pour appeler la fonction `onWeatherInput` avec une chaîne qui sera concaténée dans l'appel Api sur App.js.

☀ **showFavoris** : celui-ci fonctionne comme un toggle, au clic, la liste des favoris est display, sinon par défaut son état est à false. Les favoris sont stockés dans le local storage, dans le composant <SuggestionBox/>

☀ **SuggestionBox** : composant qui affiche une liste de suggestions de villes et permet à l'utilisateur d'ajouter ou de supprimer des villes de ses favoris.

- Le composant utilise le hook `useState` pour gérer deux états : `favs`, qui stocke les villes favorites de l'utilisateur, et `results`, qui stocke les résultats des suggestions.

- Deux fonctions sont définies : `addToFavs` pour ajouter une ville aux favoris et `removeFromFavs` pour supprimer une ville des favoris. Ces fonctions mettent à jour à la fois l'état local et le stockage local (`localStorage`) pour conserver les favoris de l'utilisateur même après le rechargement de la page.

- Le hook `useEffect` est utilisé pour mettre à jour les suggestions chaque fois que les valeurs de `suggestions` (les suggestions reçues) ou `favs` (les favoris de l'utilisateur) changent. Les suggestions sont d'abord filtrées pour éliminer les doublons, puis triées pour placer les favoris en premier, suivis des autres suggestions.

- Les suggestions sont affichées avec leur nom de ville, et si elles ne sont pas déjà dans les favoris, un icône étoile vide est affiché pour permettre à l'utilisateur de les ajouter, sinon une étoile pleine permettant de les supprimer des favoris.

Cette partie est donc le point d'entrée de l'application, s'ensuit donc suite à ces actions, le flux des données vers les autres composants.

◆ <WeatherMeme/> & <WeatherImage/>



Pour commencer, en discutant des options pour WeatherMeme, nous avons convenu de regrouper les memes en fonction des conditions météorologiques telles que "sunny", "partly cloudy", "rainy", etc., étant donné qu'il n'est pas littéralement possible de trouver un meme spécifique pour chaque condition météorologique. Pour cela, nous avons utilisé MongoDB pour stocker nos collections de memes.

Ensuite, notre tâche consistait à rechercher des memes qui correspondaient au mieux aux conditions météorologiques répertoriées. Une fois cette étape achevée, Cyril a mis en place un serveur Node.js en utilisant Express, et nous avons utilisé Mongoose pour établir une connexion à notre base de données MongoDB.

{ } My Queries mometeo X +				
+ Create collection	Refresh	View	Sort by	Collection Name
memes				
Storage size: 24.58 kB	Documents: 72	Avg. document size: 149.00 B	Indexes: 1	Total index size: 36.86 kB
musiques				
Storage size: 20.48 kB	Documents: 12	Avg. document size: 114.00 B	Indexes: 1	Total index size: 36.86 kB
testapis				
Storage size: 20.48 kB	Documents: 13	Avg. document size: 142.00 B	Indexes: 1	Total index size: 20.48 kB

Ainsi, il nous fallait mettre la logique en place pour display le bon meme pour la condition météo renvoyée par l'Api. WeatherApi renvoyait dans l'objet, une propriété "current", qui renfermait elle-même une propriété "condition" qui est un aussi un objet lui-même et voilà ce qui nous y intéressait :

```
1  { "current": {
2    "last_updated_epoch": 1712928600,
3    "last_updated": "2024-04-12 15:38",
4    "temp_c": 21.0,
5    "temp_f": 69.8,
6    "is_day": 1,
7    "condition": {
8      "text": "Partly cloudy",
9      "icon": "//cdn.weatherapi.com/weather/64x64/day/116.png",
10     "code": 1003
11   },
12 }
```

```
1  const weatherConditionsGroup = {
2
3    'Sunny': {
4      meme: 'sun',
5      sound: 'sun',
6      background: 'sun-background'
7    },
8    'Clear': {
9      meme: 'sun',
10     sound: 'sun',
11     background: 'sun-background'
12   },
13   'Partly cloudy': {
14     meme: 'partlycloudy',
15     sound: 'cloudy',
16     background: 'cloudy-background'
17   },
18   'Cloudy': {
19     meme: 'cloudy',
20     sound: 'cloudy',
21     background: 'cloudy-background'
22   },
23   'Overcast': {
24     meme: 'cloudy',
25     sound: 'cloudy',
26     background: 'cloudy-background'
27   },
28   'Patchy rain possible': {
29     meme: 'rain',
30     sound: 'rain',
31     background: 'rain-background'
32   },
33   'Moderate or heavy freezing rain': {
34     meme: 'rain',
35     sound: 'rain',
36     background: 'rain-background'
37   },
38 }
```

Ainsi, le composant WeatherMeme devait consommer la valeur de ``current.condition.text`` afin de la comparer avec la clé "name" stockée dans notre base de données MongoDB.

Nous avons consulté la documentation de l'API, qui fournissait une liste exhaustive des conditions météorologiques au format JSON. J'ai récupéré cette liste et l'ai modifiée pour qu'elle puisse être utilisée comme notre WeatherConditionGroup. Chaque clé de cet objet représente une condition météorologique, et chaque valeur associée à cette clé est à nouveau un objet contenant trois propriétés : ``meme``, ``sound``, et ``background``. Ainsi, chaque clé-valeur dans cet objet représente une relation entre une condition météorologique et les données associées à cette condition.

Ensuite, Cyril a mis en place l'appel vers notre API, et nous avons réfléchi à la manière de conditionner le rendu :

Nous avons procédé en mappant notre groupe de conditions, filtrant les memes selon que le nom corresponde à `currentWeatherText`. Ensuite, pour éviter que l'utilisateur ne voie le même meme plusieurs fois lorsqu'il consulte les conditions météorologiques, nous avons effectué une sélection aléatoire parmi les memes filtrés.

```
1  useEffect(() => {
2    const weatherConditionsMap = WeatherConditionsGroup; //WeatherMemeMap => données de
3    WeatherConditionsGroup
4    if (currentWeatherText && memes.length > 0) {
5      const filteredMemes = memes.filter(meme => {
6        const memeName = weatherConditionsMap[currentWeatherText];
7        return memeName && meme.name.toLowerCase() === memeName.meme;
8      });
9
10     if (filteredMemes.length > 0) {
11       const randomIndex = Math.floor(Math.random() * filteredMemes.length);
12       const randomMeme = filteredMemes[randomIndex];
13       setSelectedMeme(randomMeme);
14     } else {
15       setSelectedMeme(null);
16     }
17   } else {
18     setSelectedMeme(null);
19   }
20 }, [currentWeatherText, memes]);
```

Pour les composants WeatherImage et WeatherIcon, nous avons choisi d'utiliser un ensemble d'icônes personnalisées afin d'assurer une cohérence visuelle et une esthétique épurée dans notre application. Nous avons suivi une approche similaire à celle utilisée pour les paramètres, cette fois en nous basant sur la propriété `current.condition.code`.

WeatherAPI fournit un fichier JSON regroupant toutes les icônes disponibles ainsi que leurs codes associés. Nous avons donc récupéré ce fichier pour y intégrer nos propres icônes. Les composants WeatherImage et WeatherIcon prennent en paramètre la valeur de `current.condition.code`. Ensuite, nous utilisons cette valeur pour mapper notre tableau d'images ou d'icônes, afin d'afficher celle dont le code correspond exactement à celui renvoyé par l'API météo.

Une subtilité supplémentaire était la distinction entre les icônes pour le jour et pour la nuit. Pour résoudre cela, nous avons mis en place une logique de ternaire pour sélectionner l'image ou l'icône appropriée en fonction du code météorologique ainsi que de la valeur de `is_day` renvoyée par l'API météo.


```

1 const WeatherImage = ({ currentWeather }) => {
2   const currentCode = currentWeather?.current?.condition?.code;
3   const isDay = currentWeather?.current?.is_day === 1;
4
5   const image = weatherImages.find(image => image.code === currentCode);
6   if (!image) return null;
7
8   const selectedImage = isDay ? image.dayDisplay : image.nightDisplay;
9
10  return <div className="images-display">
11    <img className="image-display" src={selectedImage} alt="actual weather"
12  /> </div>;
13 };

```

◈ <Carousel/>

Pour enrichir l'expérience utilisateur et offrir un accès facile à une variété d'informations météorologiques complémentaires, nous avons utilisé la bibliothèque d'interface utilisateur AntDesign pour implémenter un carousel dans notre application météo. Ce carrousel permet d'afficher une gamme de contenus météorologiques supplémentaires, tels que les prévisions sur 5 jours, les prévisions heure par heure (durant 24h) et les prévisions de pluie heure par heure (24h également). Angèle s'en est occupée, et pour ma part j'ai proposé d'afficher de façon différente les prévisions de pluie, toujours dans le sens du dynamisme : nous avons le pourcentage de chance de pluie pour chaque heure. Pour ce côté visuel et de jeu avec les données, j'ai créé un composant RainDrop, une simple div en forme de goutte et son remplissage couleur se ferait en fonction du % de pluie annoncé, qu'il prendrait en props.



```

1 import React from 'react';
2 import '../stylesheet/Root.scss';
3
4 // ce composant va permettre d'afficher la troisième page du carousel
5 // avec les précipitations heure par heure de la journée
6
7 const RainDrop = ({ pourcentage }) => {
8   // Le style dynamique pour le remplissage de la goutte
9   //sur le scss la height de la div en absolute n'est pas défini
10  // pour être défini ici = remplissage
11  //pourcentage est récupéré sur app => hour.chance_of_rain = 89% de l'api météo
12  const styleRemplissage = {
13    //alors height = 89%
14    height: `${pourcentage}%`
15  };
16
17  // "no-rain" si le pourcentage est égal à 0
18  const className = pourcentage === 0 ? 'no-rain' : '';
19
20  return (
21    <div className={`rain-drop ${className}`}>
22      <div className="rain-drop-inner">
23        <div className="filled-raindrop" style={styleRemplissage}>
24          <span>{pourcentage}%</span>
25        </div>
26      </div>
27    </div>
28  );
29 };
30
31 export default RainDrop;

```

◈ <Modal/>

De l'autre côté, pour les prévisions sur 5 jours, j'ai développé une modal qui permettait, lors du clic sur un jour spécifique, d'afficher un aperçu des informations principales. Cette modal utilise un composant DetailCard réutilisable (voir Annexes) pour afficher ces détails de manière claire et concise.

Le composant Modal prend deux props : `onClose`, qui est appelé

```
const Modal = ({ onClose, dayInfo }) => {
  //choix des infos que l'on veut mettre dans la modale (dayInfo étant la récup api sur app)
  const { date, maxTemp, minTemp, rain, wind, avgtemp_c, avghumidity, uv, sunrise, sunset, wind_dir, wind_dir_text } =
    dayInfo;
  return (
    <div className="modal">
      <div className="modal-content">
        <span className="close" onClick={onClose}><IoIosCloseCircle /></span>
        <p className="modal-date">{date}</p>
        <div className="modal-part">
          <DetailCard iconSrc={sunriseIcon} description="" value={` ${sunrise}`} />
          <DetailCard iconSrc={sunsetIcon} description="" value={` ${sunset}`} />
          <DetailCard iconSrc={temp_max} description="Max" value={` ${maxTemp} °C`} />
          <DetailCard iconSrc={temp_min} description="Min" value={` ${minTemp} °C`} />
          <DetailCard iconSrc={feelsLikeIcon} description="Moy." value={` ${avgtemp_c} °C`} />
        </div>
        <div className="modal-part">
          <DetailCard iconSrc={rainIcon} description="" value={` ${rain} mm`} />
          <span className="wind-combo">
            <DetailCard iconSrc={windAnim} description="Moy" value={` ${wind} km/h`} />
            <WindIcon direction={wind_dir} wind_dir={wind_dir_text} />
          </span>
          <DetailCard iconSrc={humidityIcon} description="" value={` ${avghumidity} %`} />
          <DetailCard iconSrc={uvIcon} description="" value={` Indice ${uv}`} />
        </div>
      </div>
    </div>
  );
};

export default Modal;
```

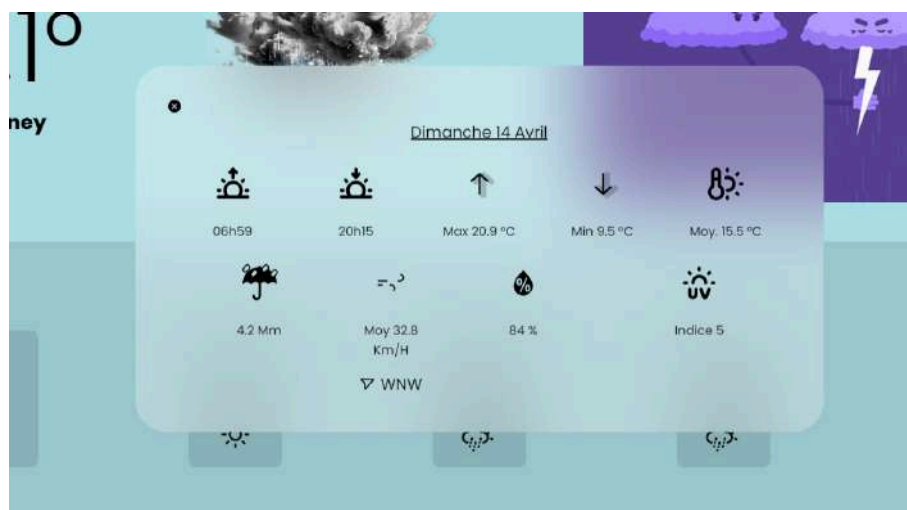
lors du clic pour fermer la modale, et `dayInfo`, qui est déstructuré. En effet, les informations du jour sont récupérées à partir du gros objet principal `currentWeather`, qui est la réponse de l'API. Lors du clic sur le composant Week, cette modal doit cibler les points que nous voulons mettre en avant et les afficher de manière dynamique. Le code ci dessous démontre la démarche :

```

// Au clic sur un des jours (props day) de prévisions dans le Carousel, la modal apparaît avec le résumé des prévisions pour ce jour
const handleDayClick = (day) => {
  const date = format(day.date, "eeee dd LLLL", { locale: fr });
  //console.log(date);
  const sunrise = hourConvert(day.astro.sunrise);
  // console.log(day.astro.sunset); //06:16 PM
  const sunset = hourConvert(day.astro.sunset);
  const maxTemp = day.day.maxtemp_c;
  const minTemp = day.day.mintemp_c;
  const rain = day.day.totalprecip_mm;
  const wind = day.day.maxwind_kph;
  const wind_dir = weatherData?.current?.wind_dir;
  const wind_dir_text = weatherData?.current?.wind_dir;
  const avgtemp_c = day.day.avgtemp_c;
  const avghumidity = day.day.avghumidity;
  const uv = day.day.uv;
  setSelectedDayInfo({ date, sunrise, sunset, maxTemp, minTemp, rain, wind, wind_dir, wind_dir_text, avgtemp_c, avghumidity, uv });
};
const handleCloseModal = () => {
  setSelectedDayInfo(null);
};

```

La première étape consistait donc à cibler le jour. Une fois le jour sélectionné, j'ai pu ensuite accéder aux propriétés qui nous intéressaient. Ensuite, j'ai mis à jour l'état de la modal pour que l'affichage dynamique puisse se faire. Ainsi, à chaque clic sur un jour de prévision, l'état du composant Modal lui-même changeait, permettant ainsi une mise à jour dynamique de son contenu.

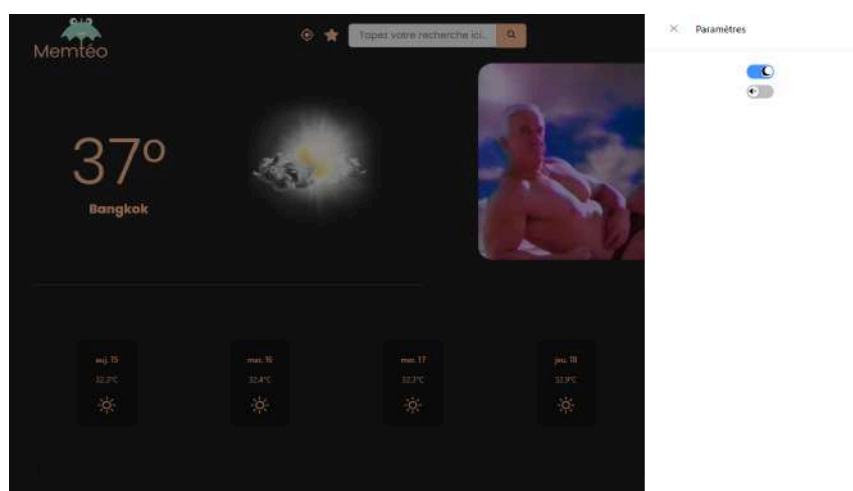
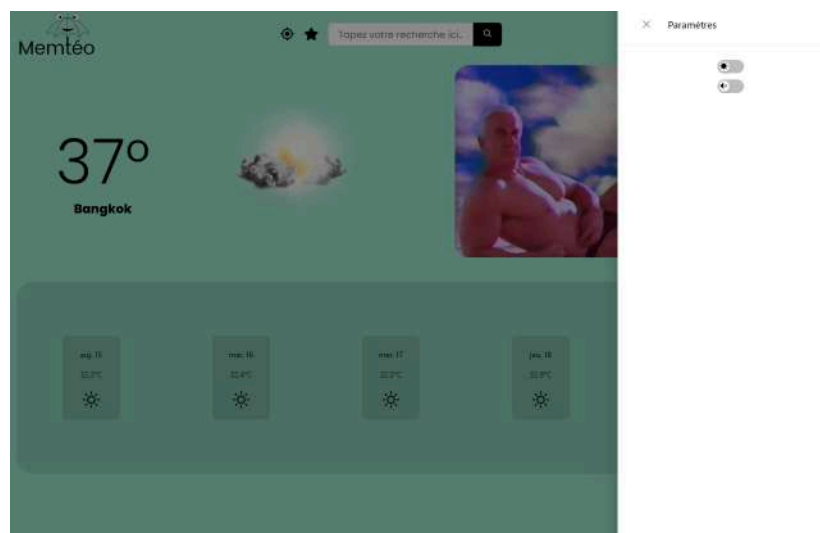


◈ <Drawer/>

Pour offrir une expérience utilisateur personnalisée, nous avons intégré le composant Drawer d'AntDesign, servant de panneau de paramètres, permettant aux utilisateurs de modifier des options telles que le DarkMode et les paramètres

sonores, grâce aux composants Switch de la même bibliothèque. Grâce à cette fonctionnalité, les utilisateurs ont la possibilité de personnaliser leur expérience météo selon leurs préférences individuelles, améliorant ainsi leur confort d'utilisation et leur satisfaction globale.

```
1  {/* Affichage des params */}
2  <div className="settings"><FiSettings className="settings-icon" onClick={() => showDrawer("right")} /></div>
3  <Drawer title="Paramètres" placement={placement} onClose={onClose} open={open}>
4    <div className="icon-display">
5      <Switch checked={isDarkMode} onClick={toggleDarkMode} className={isDarkMode ? "darkmode-switch" : "lightmode-switch"} />
6    </div>
7    <div className="sound-display">
8      <Switch checked={!isMuted} onClick={toggleMute} className={isMuted ? "muted-switch" : "unmuted-switch"} />
9    </div>
10 </Drawer>
```



◈ <WeatherIcon/>

WeatherAPI offrait la possibilité d'utiliser leurs propres icônes météorologiques. Cependant, dans un souci de cohérence visuelle et d'homogénéité, nous avons préféré utiliser notre propre ensemble d'icônes.

Pour cela, j'ai récupéré le fichier JSON mis à disposition dans leur documentation et je l'ai reconstruit pour notre projet. Chaque icône était référencée par son code. Le choix du set d'icônes a été crucial, et j'ai opté pour React Icons, ce qui nous a permis de ne pas avoir à stocker chaque SVG dans notre dossier de projet.

Pour suivre la même logique, j'ai donc remplacé chaque item par les icônes que nous avons sélectionnées. Une subtilité à prendre en compte était la nécessité d'avoir une icône pour le jour et une pour la nuit (puisque "Sunny" signifie "Clear" la nuit 😊). Une fois ce tableau de données rempli, j'ai pu passer à la création du composant WeatherIcon et à sa mise en conditionnement.

```
//récup des données json pour les icones météo (copie du tableau de
//l'api en gardant les codes des conditions météo,
//afin de dispatcher notre propre set d'icônes)
const weatherIcons = [
  {
    "code": 1000,
    "day": "Sunny",
    "icon": <WiDaySunny />,
    "nightIcon": <WiNightClear />,
  },
  {
    "code": 1003,
    "day": "Partly cloudy",
    "icon": <WiCloudy />,
    "nightIcon": <WiNightAltPartlyCloudy />,
  },
  {
    "code": 1006,
    "day": "Cloudy",
    "icon": <WiCloudy />,
    "nightIcon": <WiNightAltPartlyCloudy />,
  },
  {
    "code": 1009,
    "day": "Overcast",
    "icon": <WiDaySunnyOvercast />,
    "nightIcon": <WiNightAltPartlyCloudy />,
  },
  {
    "code": 1030,
    "day": "Mist",
    "icon": <WiDust />,
    "nightIcon": <WiDust />,
  },
  {
    "code": 1063,
    "day": "Patchy rain possible",
    "icon": <WiDayRainMix />,
    "nightIcon": <WiNightAltRainMix />,
  },
  {
    "code": 1066,
    "day": "Patchy snow possible",
    "icon": <WiDaySnow />,
    "nightIcon": <WiNightAltSnow />,
  },
  {
    "code": 1069,
    "day": "Patchy sleet possible",
    "icon": <WiDaySleet />,
    "nightIcon": <WiNightAltSleet />,
  },
]
```


Celui-ci avait donc besoin de **currentWeather.current.condition.code** et de **currentWeather.current.is_day** que je lui ai passé en props. Ainsi, feeder sur App il pouvait display le bon icône qu'il fasse jour ou nuit. Voici comment j'ai implémenté ceci :

```
const WeatherIcon = ({ code, isDay }) => {  
  // Recherche l'icône correspondant au code de condition météo dans weatherIcons ci dessus  
  const iconData = weatherIcons.find(icon => icon.code === code);  
  
  //on s'assure que l'icone est trouvée  
  if (!iconData) return null;  
  
  //selection de l'icone en fonction de Day ou Night  
  const selectedIcon = isDay ? iconData.icon : iconData.nightIcon;  
  
  //on retourne l'icone sélectionnée selon le code des conditions météo de l'api  
  return <div className="icon-display">{selectedIcon}</div>;  
};  
  
export default WeatherIcon;
```

Dans le composant WeatherIcon, nous recherchons dans notre tableau weatherIcons celui dont le code correspond strictement à celui renvoyé par l'API météo. Une fois trouvé, nous vérifions si l'API renvoie is_day = true (soit 1 par l'API).

Si c'est le cas, nous utilisons l'icône pour le jour, sinon nous utilisons l'icône pour la nuit. Ensuite, nous retournons dans le JSX une image en lui attribuant l'icône sélectionnée.

Ce composant sera principalement utilisé à l'intérieur du composant Carousel dans les composants Week (pour les prévisions sur 5 jours) et Day (pour les prévisions heure par heure).





Rent Up



Contexte du Projet



Le projet RentUp, réalisé dans le cadre du stage de fin de formation, s'inscrit dans le domaine de la location immobilière. L'objectif principal était de développer une plateforme web permettant la gestion efficace et conviviale de locations de biens immobiliers.

Cette plateforme devait offrir des fonctionnalités telles que l'ajout des biens à louer, la gestion des locataires et des quittances de loyer.

Le projet visait à mettre en pratique les compétences acquises durant la formation, notamment en développement web full-stack, en utilisant des technologies telles que PHP, JavaScript, et MySQL.

Le contexte du projet impliquait une collaboration étroite avec l'équipe encadrante pour répondre aux besoins spécifiques des utilisateurs et assurer le déploiement réussi de la plateforme dans un environnement professionnel.

L'Équipe



✱ Slamup est une jeune entreprise, elle a fêté ses 1 an il y a peu. Amaury Kozak l'a racheté après avoir travaillé pour Cocotte en Papier, qui à la base comptait 80% de print, et 20% de web. La tendance s'est vite inversée et Amaury l'a reprise. Depuis, il travaille avec un alternant Anthony qui est présent chez Slamup depuis un an et demi. Ils comptent aujourd'hui de jolis projets :

- ❖ Centre Oscar Lambret : Septembre en Or Design et développement d'un site événementiel afin de sensibiliser aux cancers pédiatriques. Faites du sport en soutien et convertissez vos efforts physiques en don au profit de la recherche.



- ❖ HV Développement : Refonte graphique et technique de leur e-commerce afin de l'internationaliser. Création d'une API centralisant les commandes de toutes leurs plateformes de vente. Développement d'un extranet et automatisation de leur flux de production.

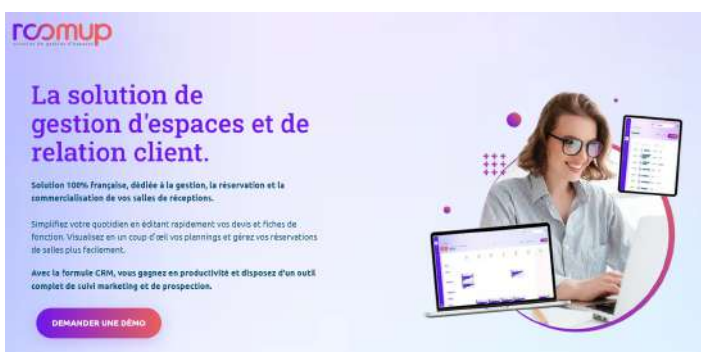


❖ Silver Home Services :

Développement d'une web app dédiée à la gestion du quotidien d'une entreprise de services à domicile : prestations, comptabilité, RH, avance immédiate Urssaf, planification, etc...



- ❖ Roomup : un Saas qui est utilisé aujourd'hui par 8 hôtels:
 - Réservation de salles
 - Proposition commerciale en 2mn
 - Module CRM : Connexion des utilisateurs Microsoft 365, Synchronisation automatique des e-mails et agendas des collaborateurs
 - Pilotage et suivi des dossiers
- ❖ Prospection commerciale (Gestion des campagnes e-mailing et phoning)
- ❖ Reporting (temps réel sur les performances, export du reporting, gestion des budgets prévisionnels)



Contraintes et Livrables



La contrainte principale était le temps imparti pour le stage. En seulement quatre semaines, Amaury m'a présenté un projet prometteur qui couvrait un large éventail de tâches. Dès le début, il était clair que la découverte des projets existants, la compréhension de leur architecture et l'estimation du temps nécessaire pour chaque aspect seraient essentielles.

Amaury m'a toujours rappelé que malgré le délai serré, il s'agissait avant tout d'une opportunité d'acquérir de l'expérience en milieu professionnel. Il a souligné à plusieurs reprises qu'il n'y avait pas de pression excessive, mais plutôt l'occasion de se développer. J'avais expressément demandé à être poussé hors de ma zone de confort, et dans les moments difficiles, Amaury s'est montré très disponible, bienveillant et pédagogue.

Chaque jour, nous prenions le temps de planifier notre journée et de revoir les progrès réalisés. Ces échanges étaient extrêmement enrichissants. Bénéficier des conseils d'un tuteur avec quinze ans d'expérience dans le développement informatique a considérablement élargi ma vision du code et des solutions possibles.

En ce qui concerne les livrables, j'ai accompli ma mission avec succès deux jours avant la fin du stage. De plus, Amaury m'a confié un projet annexe de création de maquette pour le Centre Oscar Lambret, dans le cadre de Septembre en Or 2024. Récemment, il m'a informé que le COL avait validé ma proposition.

Dans l'ensemble, je considère que les contraintes et les objectifs de livraison pour mon projet chez Slamup ont été non seulement respectés, mais même dépassés.

Environnement Technique

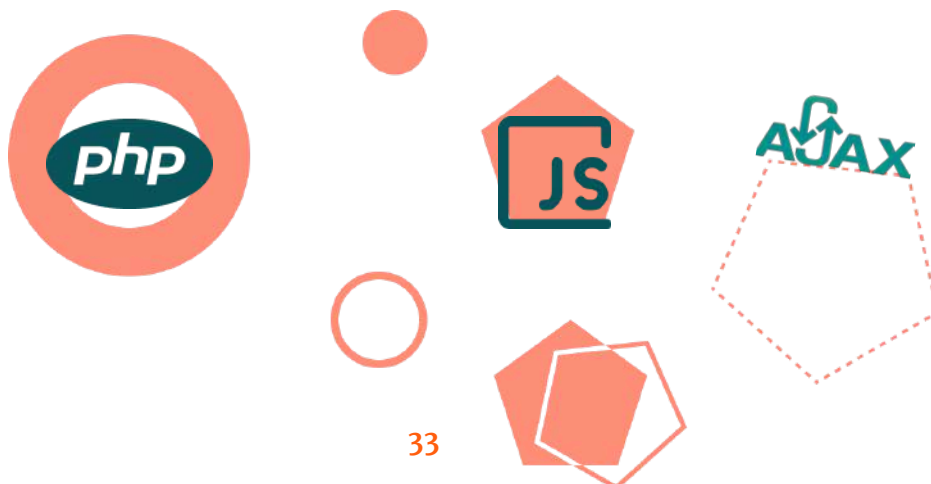


Le projet Rentup s'est révélé être une expérience full-stack, impliquant une compréhension approfondie de leur architecture en PHP orienté objet, ainsi que de JavaScript (notamment JQuery et AJAX).

Pour moi, l'utilisation de requêtes AJAX était une nouveauté. Bien que j'aie déjà des connaissances en PHP et JavaScript grâce à nos cours précédents, le PHP restait relativement éloigné de mes compétences. Cependant, j'ai réussi à m'adapter et à utiliser mes connaissances existantes, notamment en ayant travaillé avec des frameworks comme Laravel lors de nos cours précédents. J'ai particulièrement apprécié découvrir leur approche faite-maison, ainsi que leur framework CSS, développé par Amaury au cours de sa carrière et qu'il maîtrise parfaitement.

Au cours de ce projet, j'ai également eu l'opportunité de découvrir des outils tels que Looping MCD et WinSCP, utilisés respectivement pour la gestion de la base de données et le déploiement de l'application sur le serveur Slamup.

Bien que ce projet ait couvert un large éventail de compétences nécessaires pour le titre, j'ai choisi de me concentrer principalement sur la partie Backend. Cela m'a permis de mettre en pratique mes connaissances et de développer un projet de A à Z, en confrontant mes compétences à des défis concrets.



Compétences couvertes par le projet



◆ Mettre en place une BDD Relationnelle

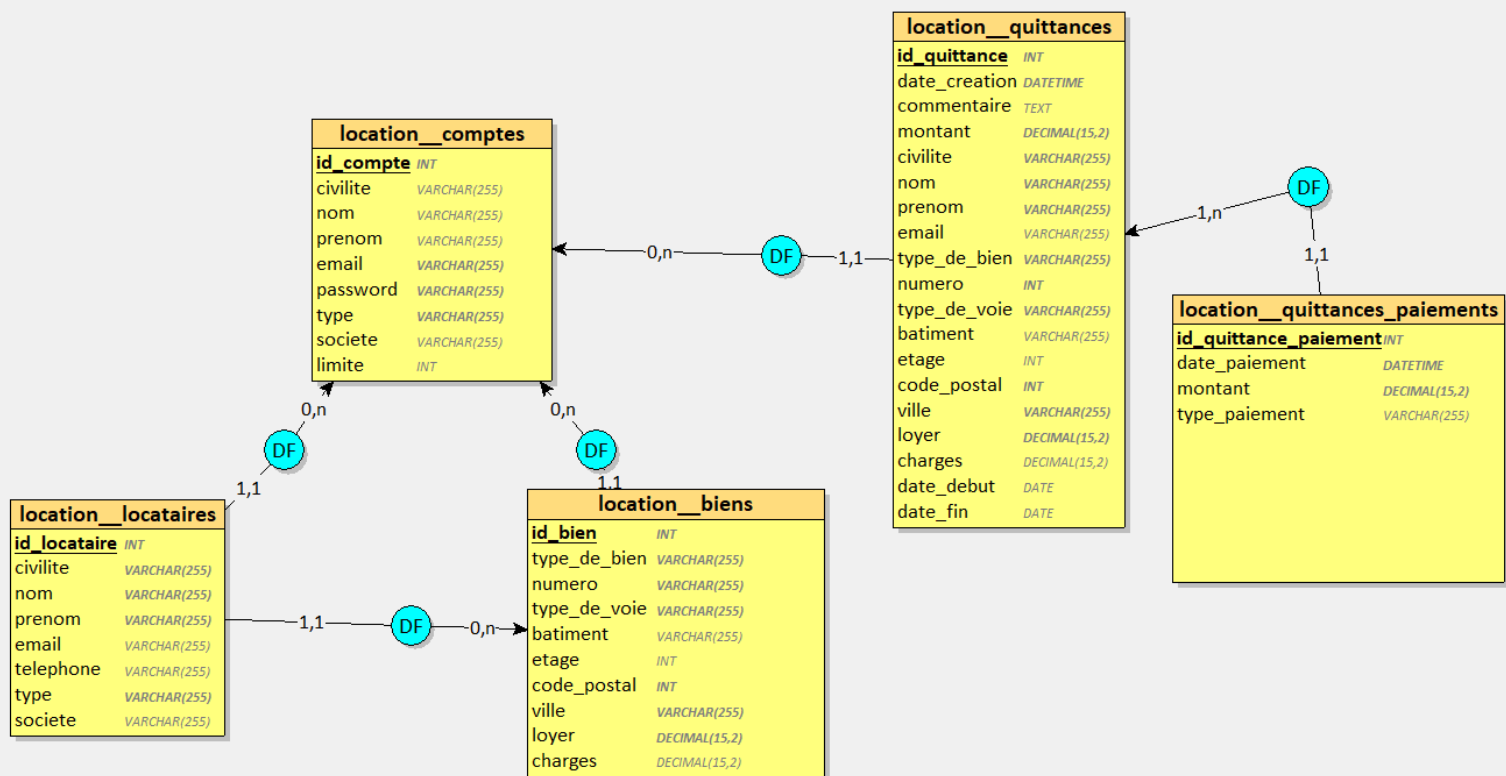


Pour mettre en place une base de données relationnelle, j'ai utilisé Looping MCD pour conceptualiser le schéma de la base de données. À partir de cette conceptualisation, j'ai extrait le script SQL nécessaire pour créer les tables et les relations entre elles. Ensuite, j'ai lancé ce script sur phpMyAdmin après avoir créé la base de données "location".

Cette base de données, nommée "location", contient plusieurs modules que j'ai développés, notamment les tables suivantes :

1. Comptes : Cette table stocke les informations relatives aux comptes des utilisateurs, telles que les identifiants, les adresses e-mail et les mots de passe.
2. Biens : Cette table gère les informations sur les biens immobiliers, y compris les adresses, les types de biens et les propriétaires associés.
3. Locataires : Cette table enregistre les détails des locataires, tels que leurs noms, adresses et informations de contact.
4. Quittances : Cette table stocke les informations sur les quittances de loyer, telles que les montants, les dates, le locataire et le bien associés.

En mettant en place cette base de données relationnelle, j'ai pu structurer efficacement les données nécessaires à la gestion des locations immobilières, facilitant ainsi leur stockage, leur récupération et leur manipulation dans l'application.



Dans cette base de données, les différentes entités sont interconnectées par des relations clés étrangères. Par exemple, dans la table "location__biens", la clé étrangère "id_compte" fait référence à la table "location__comptes", indiquant quel compte est associé à quel bien.

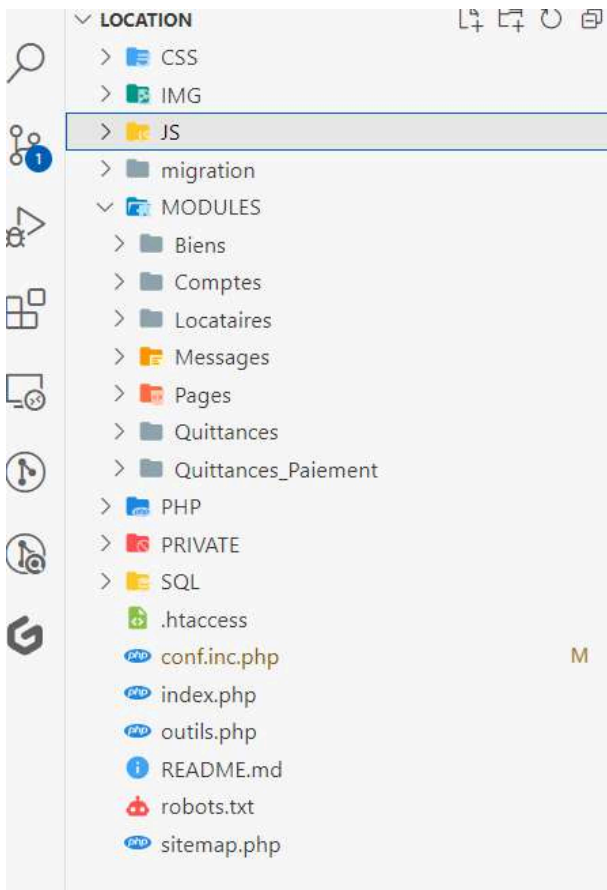
De même, dans la table "location__locataires", les clés étrangères "id_compte" et "id_bien" font référence aux tables "location__comptes" et "location__biens" respectivement, permettant de lier les locataires aux biens qu'ils occupent.

Enfin, dans la table "location__quittances", la clé étrangère "id_compte" fait référence à la table "location__comptes", assurant ainsi l'intégrité des données lors de l'enregistrement des quittances de loyer. Ces relations garantissent la cohérence et l'efficacité de la gestion des informations au sein de la base de données.

Compétences couvertes par le projet

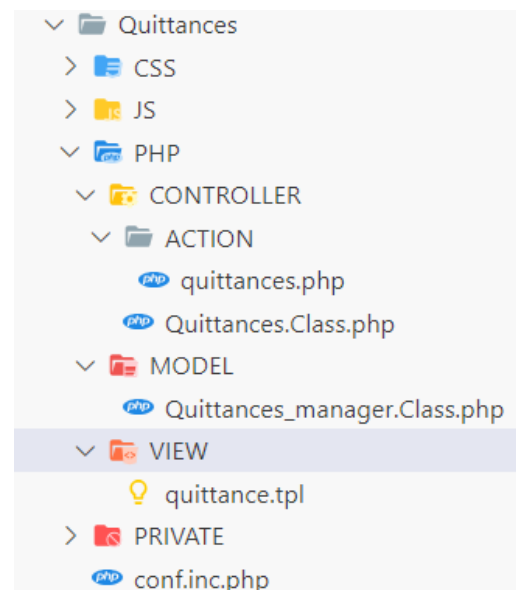


◆ Développer les composants d'accès aux données SQL



Pour développer les composants d'accès aux données SQL pour chaque module (comptes, biens, locataires, quittances) selon le modèle MVC (Modèle-Vue-Contrôleur), j'ai commencé par créer les modèles et les contrôleurs correspondants.

Pour le module des quittances, par exemple, j'ai élaboré un modèle qui définit les opérations liées aux données des quittances, telles que la création, la lecture et la suppression. Ensuite, j'ai mis en place un contrôleur qui agit comme une interface entre le modèle et la vue, gérant les actions et les interactions avec la base de données.



Pour permettre à l'utilisateur de consulter toutes les quittances, j'ai développé une fonction de lecture dans le modèle des quittances qui récupère toutes les données des quittances depuis la base de données. Cette fonction est ensuite appelée par le contrôleur des quittances pour fournir les données à la vue appropriée (box_quittances).



```

<?php
class Quittances_manager extends Db_Manager
{

    public function __construct()
    {
        parent::set_table('quittances');
        parent::set_default_id('id_quittance');
    }

    public function form_creation_quittance($id_compte, $data = null)
    {...
    }

    public function prev_quittance($id_compte, $data, $id_quittance =
null{
    ...;
    }

    public function box_quittances($id_compte = null)
    {..
    }

    public function get_quittances_par_compte($id_compte)
    {
        $quittances = $this->get_list("WHERE id_compte=" . $id_compte);
        return $quittances;
    }
}

```

Pour la création d'une quittance, j'ai développé un script PHP qui reçoit les données nécessaires depuis la vue (telles que la date de création, le montant, les informations du locataire, etc.), puis insère ces données dans la base de données en utilisant des requêtes SQL appropriées. Ce script PHP est ensuite appelé depuis la vue lorsqu'un utilisateur souhaite créer une nouvelle quittance.

```
public function form_creation_quittance($id_compte, $data = null)
{
    $biens_manager = new Biens_manager;
    $locataires_manager = new Locataires_manager;
    $id_quittance = $_GET['id_quittance'];

    $liste_biens = $biens_manager->get_list('WHERE id_compte=' . $id_compte);
    $options_value_biens = [];
    foreach ($liste_biens as $bien) {
        $options_value_biens[$bien->get_id_bien()] = $bien->get_adresse_bien();
    }

    $liste_locataires = $locataires_manager->get_list("WHERE id_compte=" . $id_compte);
    $options_value_locataires = [];
    foreach ($liste_locataires as $locataire) {
        $infos_locataire = $locataire->get_civilite() . ' ' . $locataire->get_nom() . ' ' . $locataire->get_prenom() . ' ';
        $options_value_locataires[$locataire->get_id_locataire()] = $infos_locataire;
    }

    $form = '<form id="form_creation_quittance" action="/action/Quittances/quittances" method="post" enctype="multipart/form-data"
    >';
    // $form .= '<div class="col six">';
    $form .= '<div class="box_form">';
    $form .= Formulaires::hidden('url', $_SERVER['REQUEST_URI']);
    if ($data['id_quittance']) {
        $form .= Formulaires::hidden('id_quittance', $data['id_quittance']);
    };

    $form .= Formulaires::hidden('id_compte', $id_compte);
    $form .= Formulaires::hidden('id_quittance', $id_quittance);

    $form .= Formulaires::input('date_creation', 'Création de la quittance : ', $data['date_creation'], 'date');
    $form .= Formulaires::input('date_quittance', false, $data['date_debut']);
    $form .= Formulaires::select('id_bien', 'Sélectionner un bien : ', $data['id_bien'], $options_value_biens);
    $form .= Formulaires::select('id_locataire', 'Sélectionner un locataire : ', $data['id_locataire'], $options_value_locataires);
    $form .= Formulaires::textarea('commentaire', 'Commentaire : ', $data['commentaire']);
    $form .= '</div>';
    // $form .= '</div>';
    $form .= '<div class="clear"></div>';
    $form .= '</form>';
    return $form;
}
```

En ce qui concerne la suppression d'une quittance, j'ai mis en place une fonctionnalité utilisant AJAX pour permettre à l'utilisateur de supprimer une quittance directement depuis l'interface utilisateur, sans avoir à recharger la page. Lorsque l'utilisateur clique sur le bouton de suppression, une requête AJAX est envoyée au serveur, qui exécute une requête SQL de suppression dans le modèle des quittances pour supprimer la quittance correspondante de la base de données. (Annexes)

Ces fonctionnalités ont été implémentées de manière à garantir la sécurité des données et la facilité d'utilisation pour l'utilisateur, tout en respectant les principes du modèle MVC pour assurer la maintenabilité et la scalabilité du code. Des captures d'écran seront fournies pour illustrer ces différentes étapes de développement dans les Annexes.

Évidemment, sur chaque module, le CRUD a donc été mis en place (bien, locataire, quittance).

Compétences couvertes par le projet



◆ Développer les composants métier côté serveur



Concernant cette partie j'ai choisi de présenter 2 choses :

- ★ la génération d'un pdf à la volée (des quittances).
- ★ l'envoi par mail de cette quittance au locataire.

HTML2PDF



Pour ce premier point, générer un pdf de façon dynamique, j'ai fait la découverte de la grande classe HTML2PDF. La première chose que j'ai donc dû créer était un fichier .tpl (template) pour générer la structure du pdf des quittances. Ces fichiers tpl sont donc des fichiers templates appelés par des pages d'appel en PHP : pages classiques avec des balises prédéfinies (html) qui appellent des fonctions PHP.

Dans ce fichier quittance.tpl (Annexes) j'ai aussi appris à intégrer les variables `###CIVILITE_LOCATAIRE##`, `###NOM_LOCATAIRE##` etc... dans cette syntaxe, que l'on vient déclarer dans le fichier conf.inc.php et dont la valeur sera défini dans la méthode quittance() de la class PDF.

Ensuite, pour générer ce pdf il fallait donc une action de l'utilisateur. Sur ma méthode prev_quittance() dans le Quittances_Manager, j'ai donc intégré un lien avec l'icône pdf. De là nous allons donc utilisé la méthode GET : si "pdf" est

```
//VRARIABLES PDF
define("VARIABLES_PDF", [

    '##DATECOMMENTAIREQUITTANCE##',
    '##MONTANT_QUITTANCE##',
    '##DATE_DEBUT__QUITTANCE##',
    '##DATE_FIN__QUITTANCE##',
    '##CIVILITE_COMPTE##',
    '##NOM_COMPTE##',
    '##PRENOM_COMPTE##',
    '##EMAIL_COMPTE##',
    '##TYPE_DE_BIEN##',
    '##NUMERO##',
    '##TYPE_DE_VOIE##',
    '##BATIMENT##',
    '##ETAGE##',
    '##CODE_POSTAL##',
    '##VILLE##',
    '##LOYER##',
    '##CHARGES##',
    '##CIVILITE_LOCATAIRE##',
    '##NOM_LOCATAIRE##',
    '##PRENOM_LOCATAIRE##',
    '##EMAIL_LOCATAIRE##',
    '##TELEPHONE_LOCATAIRE##'

]);
```

généré dans l'url, alors le traitement peut commencer.

Je vais schématiser ce chemin ci-après :



```
//redirection vers /pdf/numéro de la quittance
$prev .= '<a href="/pdf/' . $id_quittance . '" target="_blank"><i class="fa-solid fa-file-pdf"></i></a>';
$prev .= '</div>';
```

RewriteRule ^pdf/(.*)\$ /index.php?pdf=\$1 [L,QSA]



```
} else if (!empty($_GET['pdf'])) {
    $id_quittance = $_GET['pdf'];
    $id_compte = $_SESSION['compte']->get_id_compte();

    //cas où la quittance qui va être consultable appartient bien au compte actuellement connecté
    $quittances_manager = new Quittances_manager;
    $quittance = $quittances_manager->find('WHERE id_compte=' . $id_compte);
    $date_creation_timestamp = strtotime($quittance->get_date_creation());
    $date_creation_quittance = date("d-m-Y", $date_creation_timestamp);

    $pdf_name = 'Quittance ' . $quittance->get_nom() . $quittance->get_prenom() . '.pdf';
    $date_creation_quittance = date("d-m-Y", $date_creation_timestamp);

    if ($quittance) { //true
        $bdd_vars = [
            'id_quittance' => $id_quittance,
        ];
        $filename = strtoupper($pdf_name) . '.pdf';
        $pdf = new PDF($bdd_vars);
        $pdf->export_file($filename);
        exit;
    }
}
```



L'utilisateur clique sur le lien pour générer le pdf de la quittance. L'id_quittance est véhiculé avec la superglobale \$_GET qui va contenir ainsi cet id dans la requête via l'url. Dans notre fichier htaccess, nous avons une règle de réécriture qui dit ainsi que si sur l'url d'index.php, "pdf" est "trouvé" alors nous agissons comme cela :

1. On retrouve donc grâce à l'id de la quittance, le reste des propriétés de cette quittance.
2. On vérifie que le compte qui consulte cette quittance lui appartient (afin d'éviter qu'un utilisateur puisse saisir quelque chose dans l'url et d'accéder à des documents qui ne le concernent pas).
3. On définit le nom que l'on veut donner au pdf.
4. On instancie une nouvelle classe PDF en lui donnant l'id_quittance.
5. On utilise la méthode export_file en lui donnant le filename que l'on veut véhiculer.

La méthode export_file() prendra en paramètres le filename donc et l'output. En somme, comment nous souhaitons que ce PDF ressorte, cette classe

offre plusieurs options : celle qui nous intéresse ici pour la prévisualisation est donc l'output = "I" - output qui permet de consulter ce pdf généré dans le navigateur.

Dans cette classe PDF, une méthode que j'ai développée simplement appelée `quittance()`, permet de récupérer les informations précises pour cette quittance. Le propriétaire, le locataire, le bien en question occupé et évidemment les montants concernant ce bien. Ces propriétés sont donc reliées aux variables précédemment définies dans `conf.inc.php` et ainsi dynamiquement le pdf sera rempli selon toutes ces informations.

1 / 1 | - 70% + |  

PROPRIETAIRE	LOCATAIRE
M. Groot Jeremie	M. Duynslaeger Sacha
groot@gmail.com	152 rue Nationale (Appartement)
	Bâtiment : A Etage : 2
	59150 Croix

Je soussigné M. Groot Jeremie propriétaire du logement désigné ci-dessus, déclare avoir reçu de M. Duynslaeger Sacha, la somme de 530.00 euros, au titre du paiement du loyer et des charges pour la période de location du au et lui en donne quittance, sous réserve de tous mes droits.

Libellé	Montant (€)
Loyer	380.00 €
Charges	100.00 €
Total	530.00 €

Commentaire :

A la date du 27-03-2024

Cette quittance annule tous les reçus qui auraient pu être établis précédemment en cas de paiement partiel du montant du présent terme. Elle est à conserver pendant trois ans par le locataire (loi n° 89-462 du 6 juillet 1989 : art. 7-1).



Concernant ce deuxième point, maintenant que nous avons pu générer le pdf d'une quittance de façon dynamique, l'intérêt pour l'utilisateur était donc de pouvoir l'envoyer directement en mail à son locataire.

La première chose que j'ai pu faire était donc de mettre en place un lien qui au clic puisse permettre cette action.



```
$prev .= '<p><a href="/action/comptes/envoi_mail_quittance?id_quittance=' . $id_quittance .  
'"><i class="fa-solid fa-envelope"></i></a> Envoyer : ' . $data['email'] . ' .</p>';
```

Ensuite, créer ce fichier php qui permettrait donc de traiter cette action. J'ai donc pris connaissance de cette class Email et ainsi dérouler mon action de la sorte :

- On récupère l'id de la quittance en question grâce une nouvelle fois à la superglobale \$_GET.
- On fait appel au Quittances_Manager pour retrouver toutes les informations de cette quittance.
- On fait ensuite appel au Comptes_Manager pour retrouver les informations de l'utilisateur.
- Une fois tout cela de fait, on crée une nouvelle instance de la class Email.
- on alimente le mail (sujet, destinataire, locataire etc...)
- on définit le contenu du mail
- on nomme la pièce jointe (ici : Quittance-NOMPrenom-DATE)
- on crée une nouvelle instance de la class PDF en lui donnant l'id de la quittance que l'on veut générer
- on lui donne un output différent cette fois-ci : "S", qui permet de convertir le contenu du pdf en chaîne de caractères. Nous ne voulons pas stocker le pdf sur le serveur, ainsi cette option était la plus adéquate.
- dernière étape : on configure cette nouvelle class Email avec les méthodes appropriées en settant les valeurs que l'on a défini juste avant :
 - ◆ sujet, destinataire, contenu, pièce jointe etc...

On peut ainsi effectuer l'envoi avec la fonction envoi_mail().

```

<?php
//récup id quittance pour récup les infos
$id_quittance = $_GET['id_quittance'];
$quittances_manager = new Quittances_manager;
$quittance = $quittances_manager->find('WHERE id_quittance=' . $id_quittance);
//récup des infos du compte
$id_compte = $quittance->get_id_compte();
$comptes_manager = new Comptes_manager;
$compte = $comptes_manager->find('WHERE id_compte=' . $id_compte);

////////////////////////INSTANCE EMAIL////////////////////////////////////////
$email = new Email();
//alimenter le mail
//définition du sujet du mail
$sujet = "Quittance de Loyer";
//destinataire email
$destinataire = [$quittance->get_email()];
$locataire = $quittance->get_civilite() . ' ' . $quittance->get_nom() . ' ' . $quittance->get_prenom();
$proprietaire = $compte->get_civilite() . ' ' . $compte->get_nom() . ' ' . $compte->get_prenom();
//format date
//$quittance_date_creation = 2024-04-03 00:00:00
$date_creation_timestamp = strtotime($quittance->get_date_creation());
//$date_creation_timestamp = 1712095200
$date_creation_quittance = date("d-m-Y", $date_creation_timestamp);
//date_creation_quittance = 03-04-2024

//Affichage du mois en toute lettres dans le contenu du mail
//$quittance_date_creation = 2024-04-03 00:00:00
$date = new DateTimeFrench($quittance->get_date_creation());
$mois_quittance = $date->format('F');

//contenu du message
$contenu = "Bonjour, " . $locataire . "\n\nVeuillez trouver ci-joint votre quittance de loyer pour le mois de " . $mois_quittance . ".\n\nCordialement,\n" . $proprietaire;

$piece_jointe_pdf = 'Quittance-' . $quittance->get_nom() . ' ' . $quittance->get_prenom() . '-' . $date_creation_quittance;
$bdd_vars = [
    'id_quittance' => $id_quittance,
];
$filename = strtoupper($piece_jointe_pdf) . '.pdf';
$pdf = new PDF($bdd_vars);
$contenu_piece_jointe = $pdf->export_file($filename, output: 'S');

// Configurer l'e-mail
$email->set_sujet($sujet);
$email->set_to($destinataire);
$email->set_detail($contenu);
$email->set_bcc('');
$email->set_reply_to('');
$email->set_from($compte->get_email());
$email->set_from_name($proprietaire);
$email->set_piece_jointe([$contenu_piece_jointe => $filename]);
//contenu => nom du fichier

$envoi = $email->envoi_mail();

```

Compétences couvertes par le projet



◆ Déploiement



Enfin, sur cette dernière partie j'ai pu connaître le déploiement de 2 façons différentes sur chacun des projets.

Pour notre application Mem'téo nous avons déployé notre projet avec Vercel.

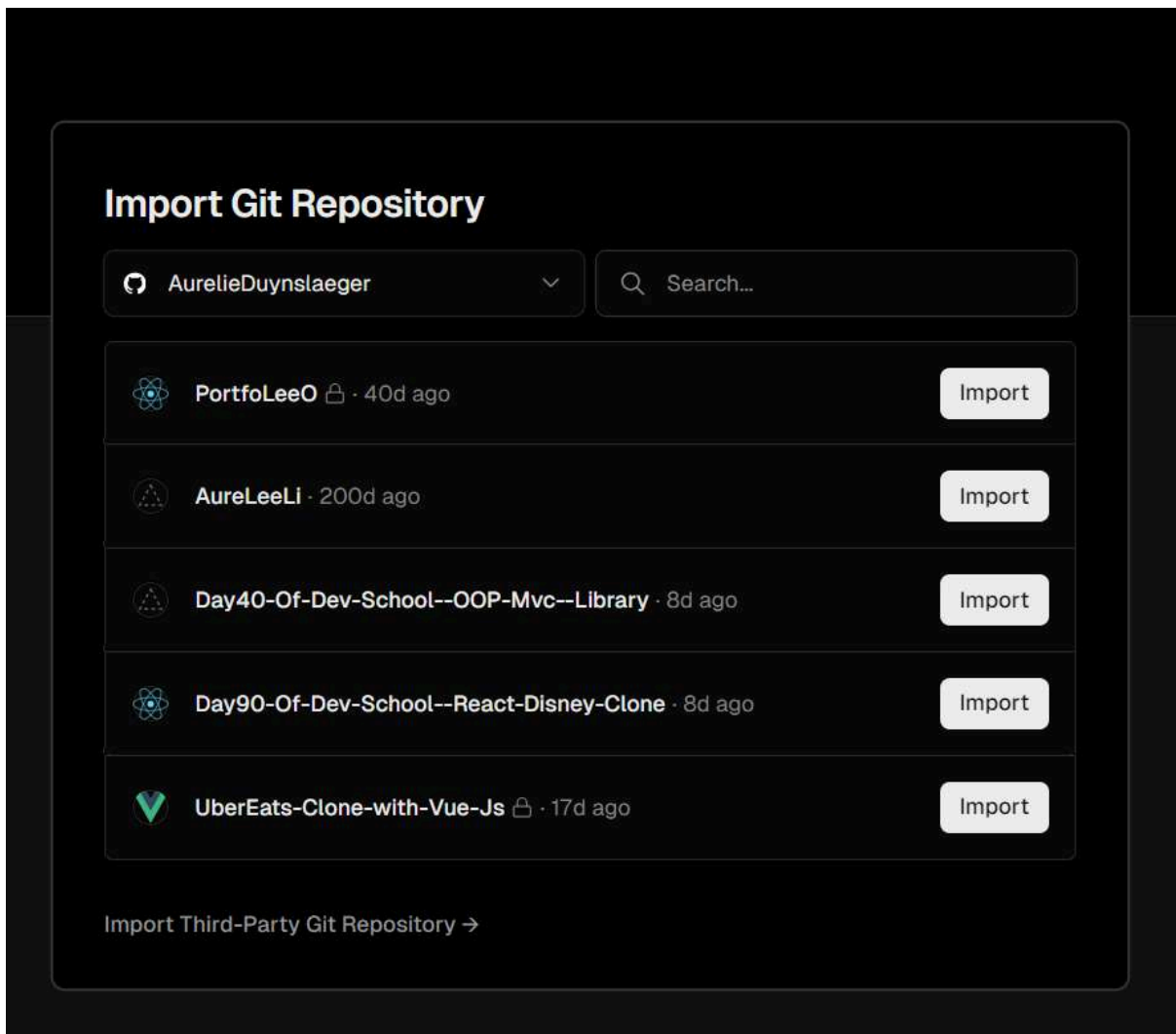
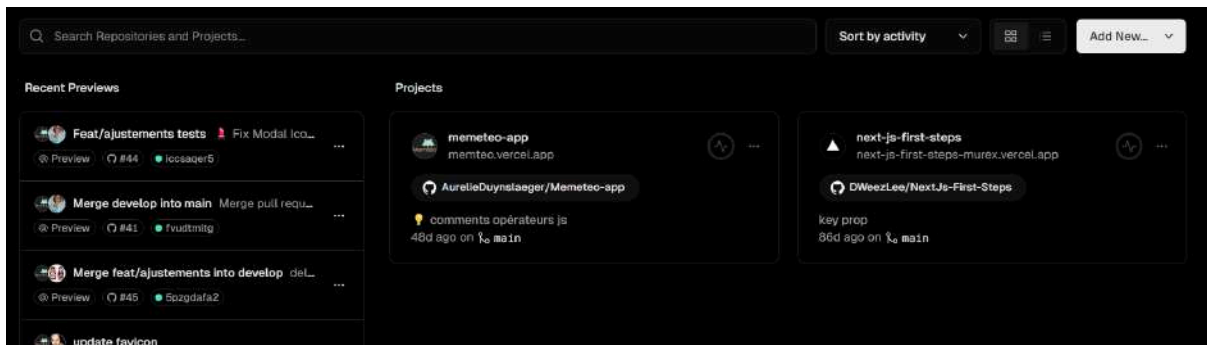


Vercel, par son interface utilisateur permet de façon très simplifiée de pouvoir déployer une application. En effet, en connectant directement le dépôt GitHub du projet, Vercel s'occupe ensuite du build process. Chaque push qui est fait sur la main, Vercel redéploie en détectant ce changement et ainsi la version hébergée de l'application est la dernière version.



Il est également possible de configurer des variables d'environnement (ce que nous avons fait notamment pour notre ApiKey que j'ai donc eu sur WeatherApi), des redirections.

De cette façon, vous avez un dashboard qui permet de voir les derniers changements sur le repo, les build logs et les projets hébergés. L'ajout d'un nouveau projet se fait de manière une nouvelle fois très simple : "Add New" et étant donné que votre compte GitHub est relié à Vercel, il est juste nécessaire de naviguer sur les repos et importer celui que vous voulez déployer.



Vercel s'inscrit dans le paradigme du CI/CD (Continuous Integration/Continuous Deployment). Il offre un ensemble d'outils pour automatiser le déploiement continu des applications web. Vercel permet une intégration en douceur du flux de travail de développement.

Concernant Rentup j'ai connu le déploiement avec WinSCP.



WinSCP

Le déploiement avec cet outil est un processus manuel de transfert de fichiers vers un serveur distant via le protocole SCP (Secure Copy Protocol).

Tout d'abord, il faut donc installer et configurer WinSCP sur l'ordinateur.

Pour déployer avec WinSCP, on établit d'abord une connexion sécurisée avec le serveur distant en spécifiant l'adresse IP et en fournissant le nom d'utilisateur et le mot de passe. Ensuite, on sélectionne (glisser-déposer) les fichiers à transférer depuis la machine locale vers le serveur distant. Une fois le transfert terminé, les fichiers sont mis à jour sur le serveur distant, permettant ainsi de déployer les changements sur votre site web ou votre application.

Bien que cette méthode offre un contrôle direct sur le processus de déploiement, elle est généralement plus laborieuse et sujette aux erreurs humaines par rapport aux solutions de CI/CD automatisées comme Vercel.



CONCLUSION



Cette formation de Développeur Web et Web mobile m'a donné les bases pour appréhender le marché du numérique, et effectuer mes premiers pas en entreprise pour mon stage chez Slamup.

Slamup a consolidé mes acquis et m'a permis de découvrir d'autres aptitudes que je souhaite creuser et mettre à profit dans mes prochains projets.

Je suis ravie et convaincue que ce bilan de compétences qui m'a amené sur cette voie fera preuve d'une belle reconversion. Reconversion d'ailleurs que je n'arrête pas en chemin, puisque j'ai été prise au sein de OpenClassRoom pour consolider et développer mes compétences un peu plus pour un titre de Concepteur et Développeur d'applications JavaScript REACT, que je débute d'ici quelques jours.

Je compte aussi terminer mon portfolio dans les jours qui arrivent afin de pouvoir avoir une présence numérique dès mes débuts.

Annexes FrontEnd

SCHEMA USER STORIE

1 - Ecran Loading App



3 - Paramétrage de l'appli



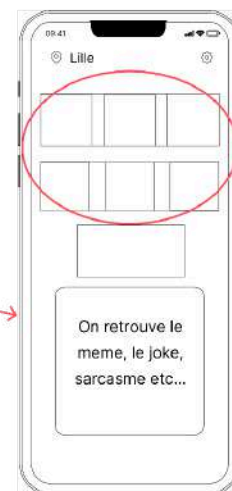
2 - Ecran Index App



5 - Recherche et Favoris



4 - Infos Détaillées



Composant a scroller :
l'utilisateur pourra, en
scrollant en haut ou en bas,
naviguer sur 3 genres
d'infos :

- prévisions heure par
heure
- prévisions sur 7 jours
- précipitations dans
l'heure

ADD DATA

EXPORT DATA

UPDATE

DELETE

```
name : "partlycloudy"
image : "https://media.giphy.com/media/8cEFp9dQCcE8M/giphy.gif?cid=790b7611dsd..."
```

```
_id: ObjectId('65df60e09035bde45591b86c')
name : "partlycloudy"
image : "https://media.giphy.com/media/3q7VFETRrjXRTaaW0o/giphy.gif?cid=ecf05e4..."
```

```
_id: ObjectId('65df60e09035bde45591b86d')
name : "partlycloudy"
image : "https://media.giphy.com/media/gmZuUqwQtllKVjNMz9/giphy.gif?cid=790b761..."
```

```
_id: ObjectId('65df60e09035bde45591b86e')
name : "cloudy"
image : "https://c.tenor.com/7BkF6U-9DecAAAAC/tenor.gif"
```

```
_id: ObjectId('65df60e09035bde45591b86f')
name : "cloudy"
image : "https://media3.giphy.com/media/v1.Y2lkPTc5MGI3NjExenZiNG85YmZxdWV4dmpq..."
```

```
_id: ObjectId('65df60e09035bde45591b870')
name : "cloudy"
image : "https://media1.tenor.com/m/vffgMZTfIjAAAAAC/cloud-nine-cloud9.gif"
```

collection meme MongoDB

Mise en évidence des names volontairement multiples pour faire un random sur les memes correspondant aux conditions météo

composant DetailCard

```
1 import React from 'react';
2
3 const DetailCard = ({ iconSrc, description, value }) => {
4   return (
5     <div className='details-card'>
6       <img src={iconSrc} alt="" width={40} height={40} />
7       <p>{description} {value}</p>
8     </div>
9   );
10 };
11
12 export default DetailCard;
```

utilisé dans la modal pour chaque informations relatives à la météo du jour (lever de soleil, coucher, vent, ressenti etc...)

composant windIcon

utilisé dans la modal pour joindre aux données de vitesse du vent, sa direction de façon plus ludique.

```
1 import React from 'react'
2 import { TbNavigation } from 'react-icons/tb';
3
4 // 0° - north wind (N)
5 // 22.5° - north-northeast wind (NNE)
6 // 45° - northeast wind (NE)
7 // 67.5° - east-northeast wind (ENE)
8 // 90° - east wind (E)
9 // 112.5° - east-southeast wind (ESE)
10 // 135° - southeast wind (SE)
11 // 157.5° - south-southeast wind (SSE)
12 // 180° - south wind (S)
13 // 202.5° - south-southwest wind (SSW)
14 // 225° - southwest wind (SW)
15 // 247.5° - west-southwest wind (WSW)
16 // 270° - west wind (W)
17 // 292.5° - west-northwest wind (WNW)
18 // 315° - northwest wind (NW)
19 // 337.5° - north-northwest wind (NNW)
20 // 360° - north wind (N)
21
22 const WindIcon = ({direction, wind_dir}) => {
23   //WindIcon récupère sur app la direction du vent de l'appel api => weatherData?.current?.wind_dir => "WSW"
24   const getIcon = (direction) => {
25     //fonction qui va trier quelle icône afficher selon la direction qu'on lui donne à consommer
26     //j'ai simplement pris une seule icône et je l'ai fait tourner selon les directions récupérés des différents vents listés ci dessus
27     switch (direction) {
28       case 'N':
29         return <TbNavigation />;
30       case 'NNE':
31         return <TbNavigation style={{ transform: 'rotate(-22.5deg)' }} />;
32       case 'NE':
33         return <TbNavigation style={{ transform: 'rotate(-45deg)' }} />;
34       case 'ENE':
35         return <TbNavigation style={{ transform: 'rotate(-67.5deg)' }} />;
36       case 'E':
37         return <TbNavigation style={{ transform: 'rotate(-90deg)' }} />;
38       case 'ESE':
39         return <TbNavigation style={{ transform: 'rotate(-112.5deg)' }} />;
40       case 'SE':
41         return <TbNavigation style={{ transform: 'rotate(-135deg)' }} />;
42       case 'SSE':
43         return <TbNavigation style={{ transform: 'rotate(-157.5deg)' }} />;
44       case 'S':
45         return <TbNavigation style={{ transform: 'rotate(-180deg)' }} />;
46       case 'SSW':
47         return <TbNavigation style={{ transform: 'rotate(-202.5deg)' }} />;
48       case 'SW':
49         return <TbNavigation style={{ transform: 'rotate(-225deg)' }} />;
50       case 'WSW':
51         return <TbNavigation style={{ transform: 'rotate(-247.5deg)' }} />;
52       case 'W':
53         return <TbNavigation style={{ transform: 'rotate(-270deg)' }} />;
54       case 'WNW':
55         return <TbNavigation style={{ transform: 'rotate(-292.5deg)' }} />;
56       case 'NW':
57         return <TbNavigation style={{ transform: 'rotate(-315deg)' }} />;
58       case 'NNW':
59         return <TbNavigation style={{ transform: 'rotate(-337.5deg)' }} />;
60       default:
61         return null;
62     }
63   }
64   return (
65     <div className='wind-icon-dir'>
66       {getIcon(direction)} {wind_dir}
67     </div>
68   )
69 }
70
71 export default WindIcon
```


Annexes BackEnd



fichier de traitement d'action sur les quittances

```
<?php
$quittance = new Quittances();
$quittances_manager = new Quittances_manager;
$locataires_manager = new Locataires_manager;
$biens_manager = new Biens_manager;
$comptes_manager = new Comptes_manager;

//récup chaîne de caractères du date picker : string '01/04/2024 - 30/04/2024'
$dates = explode(" - ", $_POST['date_quittance']);
//récup d'un tableau de dates :
// array (size=2)
//  0 => string '01/04/2024'
//  1 => string '30/04/2024'
$date_debut = $dates[0];
$date_fin = $dates[1];

if (!empty($_POST)) {
    $tab_post = [];
    $id_quittance = $_POST['id_quittance'];

    if (empty($action)) $action = !empty($id_quittance) ? "delete" : "add";

    if ($action == "add") {
        if (isset($_POST['id_compte'])) {
            $id_compte = $_POST['id_compte'];
            $compte = $comptes_manager->find_by_id($id_compte);
            if ($compte) {
                echo "Compte récupéré avec succès";
            } else {
                echo "Erreur lors de la récupération du compte";
            }
        } else {
            echo "Le champ 'id_compte' n'a pas été trouvé dans \$_POST";
        }
    }

    //on attribue les données récupérées à chaque champs de quittance
    if (!empty($_POST['id_bien'])) {
        $bien = $biens_manager->get_infos_by_id($_POST['id_bien']);
    }

    if (!empty($_POST['id_locataire'])) {
        $locataire = $locataires_manager->get_infos_by_id($_POST['id_locataire']);
    }
}
```

```

    $tab_post = [
        "date_creation" => $_POST['date_creation'],
        "date_debut" => $date_debut,
        "date_fin" => $date_fin,
        "commentaire" => $_POST['commentaire'],
        "civilite" => $locataire['civilite'],
        "nom" => $locataire['nom'],
        "prenom" => $locataire['prenom'],
        "email" => $locataire['email'],
        "type_de_bien" => $bien['type_de_bien'],
        "numero" => $bien['numero'],
        "type_de_vote" => $bien['type_de_vote'],
        "code_postal" => $bien['code_postal'],
        "ville" => $bien['ville'],
        "loyer" => $bien['loyer'],
        "charges" => $bien['charges'],
        "montant" => ($bien['loyer'] + $bien['charges']),
        "id_compte" => $id_compte
    ];

    switch ($action) {
        case 'add': {
            $id_quittance = $quittances_manager->add($tab_post);
            $_SESSION['message'] = 'Ajout d\'une quittance effectué avec succès';
            $url = "/compte/ajout-quittances?id_quittance=" . $id_quittance;
            break;
        }
        case 'delete': {
            $quittances_manager->delete('WHERE id_quittance=' . $id_quittance);
            $_SESSION['message'] = 'La quittance a bien été supprimée';
            $url = "/compte/mes-quittances";
        }
    }
    aller_url($url);
    exit;
}
```

requête SQL pour le add

```
//ACTIONS SQL
public function add($tab)
{
    $db = $this->get_db();
    $nb = count($tab);
    $i = 1;
    $sadd = "";
    $sadd2 = "";
    foreach ($tab as $key => $value) {
        $sadd .= $key;
        $sadd2 .= ":" . $key;
        if ($i < $nb) {
            $sadd .= ",";
            $sadd2 .= ",";
        }
        $i++;
    }

    $requete = $db->prepare("INSERT INTO " . PREFIXE_SQL . $this->get_table() . " (" . $sadd . ") VALUES (" . $sadd2 . ")");
    //function de debug.
    // debug_requete($requete);
    // exit;
    foreach ($tab as $key => $value) {
        $requete->bindValue(":" . $key, $value);
    }
    $requete->execute();
    // $requete->debugDumpParams();
    $last_id = $db->lastInsertId();
    return $last_id;
}
```



requête AJAX pour le delete d'une quittance

```
$('.open_dialog').off().on('click', function (e) {
    e.preventDefault();
    var element = $(this).data('element'); // quittance, ou bien, ou locataire)
    var id = $(this).data('id');
    var action = 'delete';
    var module = element + 's'; // ajoute un "s" à l'élément pour correspondre au nom du
module var url_ajax = "/" + module + "/action/" + module;

    var data_ajax = {
        action: action,
        // "id_" + element : id,
    };
    switch (element) { //bien ou loc ou quittances
        case "bien":
            data_ajax.id_bien = id;
            var message_dialog = "<p>Souhaitez-vous supprimer ce bien ?</p>";
            break;
        case "locataire":
            data_ajax.id_locataire = id;
            var message_dialog = "<p>Souhaitez-vous supprimer ce locataire ?</p>";
            break;
        case "quittance":
            data_ajax.id_quittance = id;
            var message_dialog = "<p>Souhaitez-vous supprimer cette quittance ?</p>";
            break;
        default:
            break;
    }

    var function_on_success = 'reload';

    open_dialog(message_dialog, url_ajax, data_ajax, function_on_success, null);
});
```



```
$box .= '<a class="open_dialog btn-card" data-id="' . $quittance->get_id_quittance() . '" data-element="quittance" href="#" ><i class="fas fa-trash"></i></a>';
```



méthode quittance() dans l'export_file pour feeder les variables nécessaires du pdf

```
public function quittance()
{
    $id_quittance = $this->get_id_quittance();
    $quittances_manager = new Quittances_manager;
    $tpl = file_get_contents($_SERVER['DOCUMENT_ROOT'] .
'/MODULES/Quittances/PHP/VIEW/quittance.tpl');
    $tab_method = [
        'infos_quittance',
        'infos_client',
        'infos_locataire',
        'infos_bien'
    ];
    foreach ($tab_method as $method) {
        foreach ($this->$method() as $key => $value) {
            $$key = $value;
        }
    }
    //REPLACER LES VARIABLES
    $search = VARIABLES_PDF;
    $replaceto = [
        //INFOS QUITTANCE
        $date_creation_quittance,
        $commentaire_quittance,
        $montant_quittance,
        $date_debut_quittance,
        $date_fin_quittance,
        //CLIENT
        $civilite_compte,
        $nom_compte,
        $prenom_compte,
        $email_compte,
        //BIEN
        $type_de_bien,
        $numero,
        $type_de_voie,
        $batiment,
        $etage,
        $code_postal,
        $ville,
        $loyer,
        $charges,
        //LOCATAIRE
        $civilite_locataire,
        $nom_locataire,
        $prenom_locataire,
        $email_locataire,
        $telephone_locataire,
    ];
    $html = str_replace($search, $replaceto, $tpl);
    //supprime la pagination du footer si on n'est pas dans un PDF
    // if ($obj_tpl->get_libelle() == 'Footer' && $type_file != 'PDF') {
    //     $html = str_replace('[[page_cu]]/[[page_nb]]', '', $html);
    // }

    $pages[] = $html;
    return $pages;
}
```

C:\wamp64\www\location\PHP\CONTROLLER\CLASS\Email.Class.php:263:

```
object(PHPMailer\PHPMailer\PHPMailer)[7]
  public 'Priority' => null
  public 'CharSet' => string 'UTF-8' (length=5)
  public 'ContentType' => string 'text/plain' (length=10)
  public 'Encoding' => string '8bit' (length=4)
  public 'ErrorInfo' => string '' (length=0)
  public 'From' => string 'groot@gmail.com' (length=15)
  public 'FromName' => string 'M. Groot Jeremie' (length=16)
  public 'Sender' => string 'groot@gmail.com' (length=15)
  public 'Subject' => string 'Quittance de Loyer' (length=18)
  public 'Body' => string '<!DOCTYPE html>
```

class Email instanciée

```
<html lang="fr">
```

```
<head>
```

```
    <meta charset="utf-8" />
```

```
</head>
```

```
<body>
```

```
<table border="0" cellpadding="0" cellspacing="0" width="100%"
style="font-size:12px;font-family:Arial, Verdana,
sans-serif;color:#000;">
```

```
    <tr>
```

```
        <td width="100%" align="center">
```

```
            <table border="0" cellpadding="0" cellspacing="0" width="500"
```

```
style="font-size:12px;font-family:Arial, Verdana,
sans-serif;color:#000;">
```

```
        <tr>
```

```
            <td width="500" valign="middle" align="center"><a
```

```
href="http://rentup"
```

```
style="color:#'... (length=1305)
```

```
    public 'AltBody' => string '
```

Bonjour, M. test démo idem

Veuillez trouver ci-joint votre quittance de loyer pour le mois de Avril.

Cordialement,
M. Groot Jeremie

contenu du mail

Rent Up
rentup

```
protected 'attachment' =>
```

```
    array (size=1)
```

```
        0 =>
```

```
            array (size=8)
```

```
                0 => string '%PDF-1.7
```

```
%? ? ? ?
```

```
6 0 obj
```

```
<< /Type /Page /Parent 1 0 R /LastModified (D:20240417203138+02'00') /Resources 2 0 R
```

```
endobj
```

```
7 0 obj
```

```
<</Filter /FlateDecod'... (length=7946)
```

```
    1 => string 'QUITTANCE-TEST DÉMOIDEM-17-04-2024.pdf' (length=39)
```

```
    2 => string 'QUITTANCE-TEST DÉMOIDEM-17-04-2024.pdf' (length=39)
```

```
    3 => string 'base64' (length=6)
```

```
    4 => string 'application/pdf' (length=15)
```

```
    5 => boolean true
```

```
    6 => string 'attachment' (length=10)
```

```
    7 => int 0
```

pièce jointe (string) et le nom
qu'on lui a donné via \$filename