



28 SEPTEMBRE 2022

Document présenté à

Jean-Christophe Demers

Dans le cadre du cours

420-C61-IN Projet Synthèse

PROJET SYNTHÈSE

CONCEPTION : ESCAPE ROOM MANAGER

BELONY CARLENS, GUINDON MAXENCE

CÉGEP DU VIEUX MONTRÉAL

255 Rue Ontario E, Montréal, Qc H2X 1X6




Table des matières

Maquette Interface graphique	3
Conception UML	9
Diagrammes des cas d'usages.....	9
Diagrammes des classes détaillées	13
Schéma de la structure de données externes.....	13
Éléments de conception	15
Structure de données.....	15
Patron de conception.....	17
Middleware.....	17
DAO	17
MVC.....	19
Stratégie.....	19
Expression Régulière.....	19
Algorithme	20
Mathématique	21
Outils technologiques du projet	22
Bibliographie	23

Maquette Interface graphique

The image is a wireframe of a web browser window. The browser's address bar shows the URL "www.EscapeRoommanager.com/login" and the page title is "EscapeRoom". The main content area features the heading "EscapeRoom Manager" in a large, bold font. Below this heading is a gray rectangular box containing the text "Veuillez vous connecter!". Underneath this text are four input fields: "Nom d'utilisateur", "Mot de passe", "Connection", and "Créer un compte". The "Connection" and "Créer un compte" fields are styled as buttons. The browser window includes standard navigation icons (back, forward, stop, home) and a search icon in the top left, and a double-slash icon in the bottom right corner.

EscapeRoom

www.EscapeRoommanager.com/login

EscapeRoom Manager

Veuillez vous connecter!

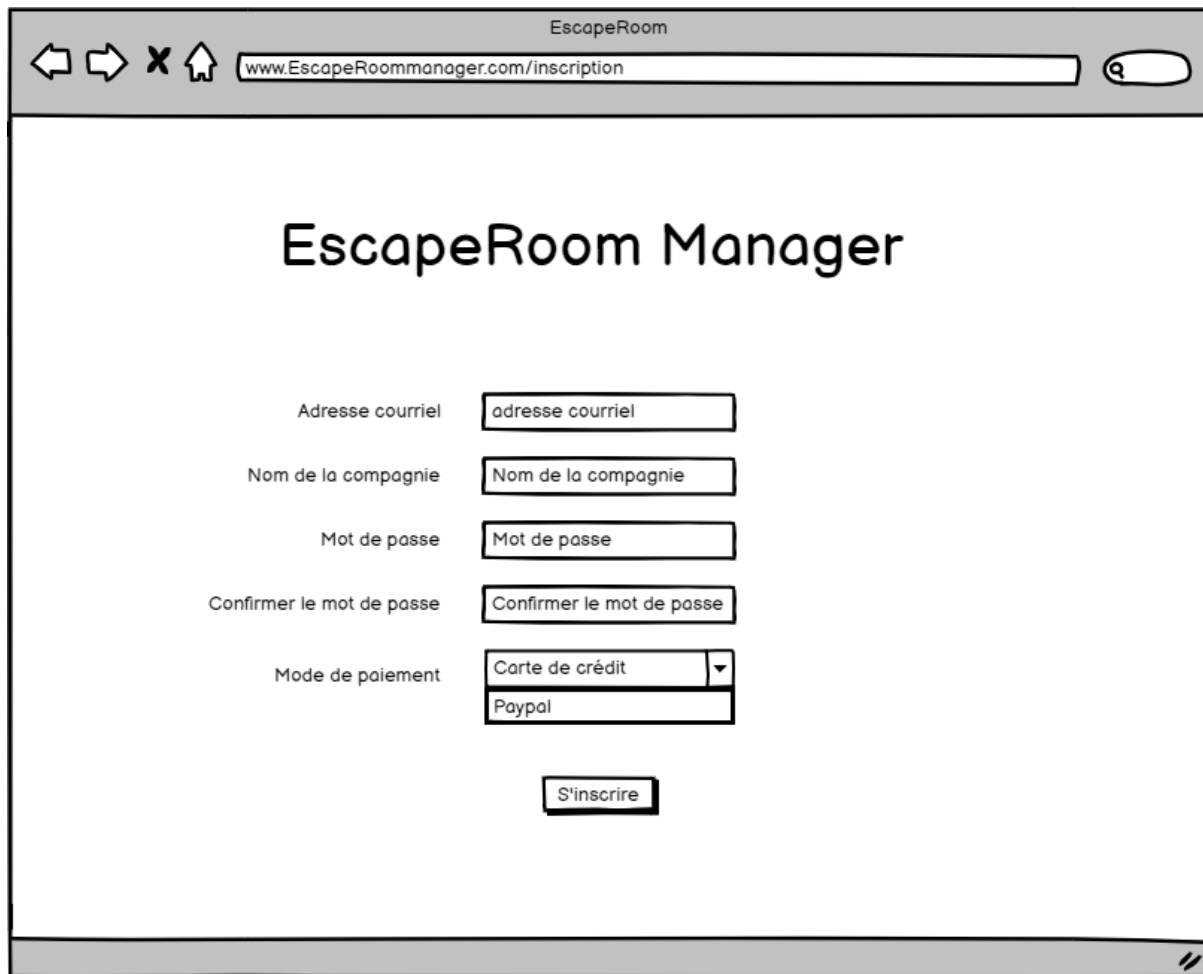
Nom d'utilisateur

Mot de passe

Connection

Créer un compte

Figure 1 Maquette de la page de connexion



EscapeRoom

www.EscapeRoommanager.com/inscription

EscapeRoom Manager

Adresse courriel

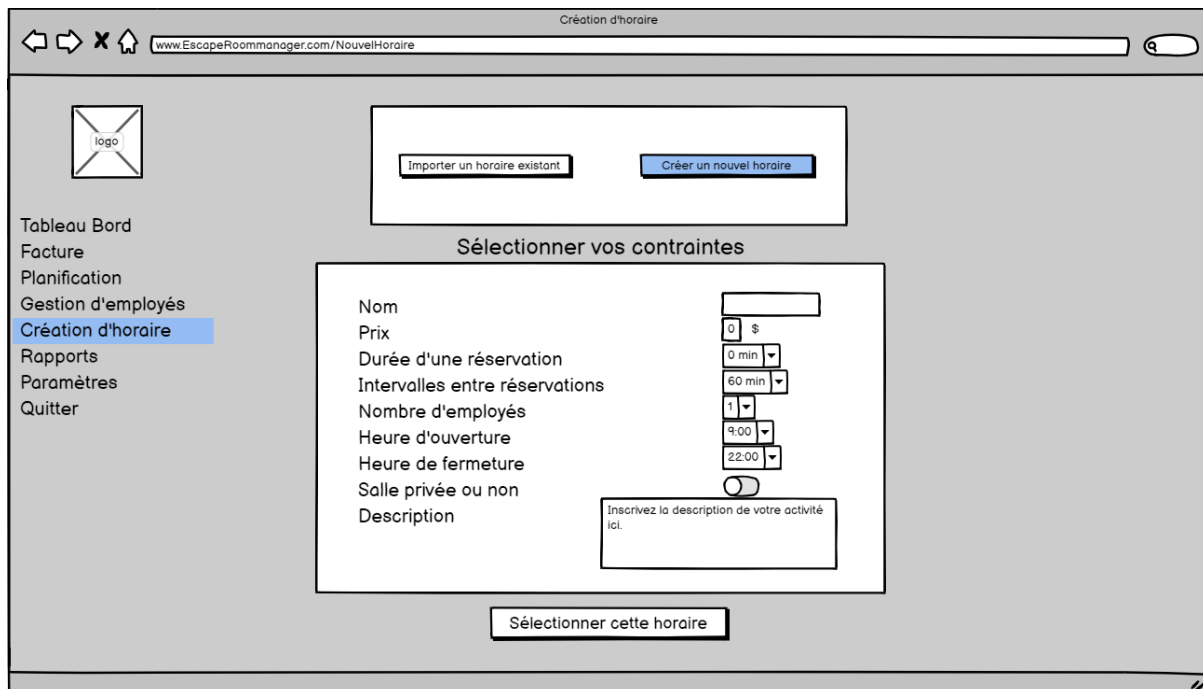
Nom de la compagnie

Mot de passe

Confirmer le mot de passe

Mode de paiement

Figure 2 Maquette de la page d'inscription



Création d'horaire

www.EscapeRoommanager.com/NouvelHoraire

logo

Tableau Bord
Facture
Planification
Gestion d'employés
Création d'horaire
Rapports
Paramètres
Quitter

Sélectionner vos contraintes

Nom

Prix \$

Durée d'une réservation

Intervalles entre réservations

Nombre d'employés

Heure d'ouverture

Heure de fermeture

Salle privée ou non ☐

Description

Figure 3 Maquette de la page pour créer un horaire

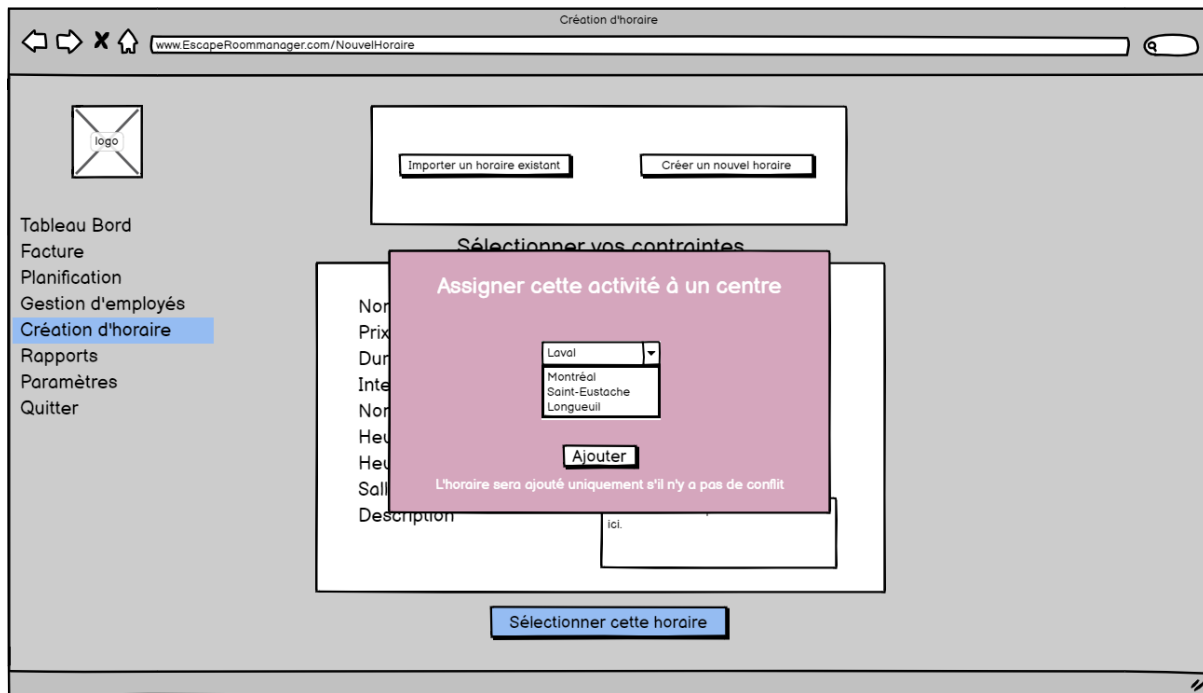


Figure 4 Maquette de la page pour créer un horaire suite

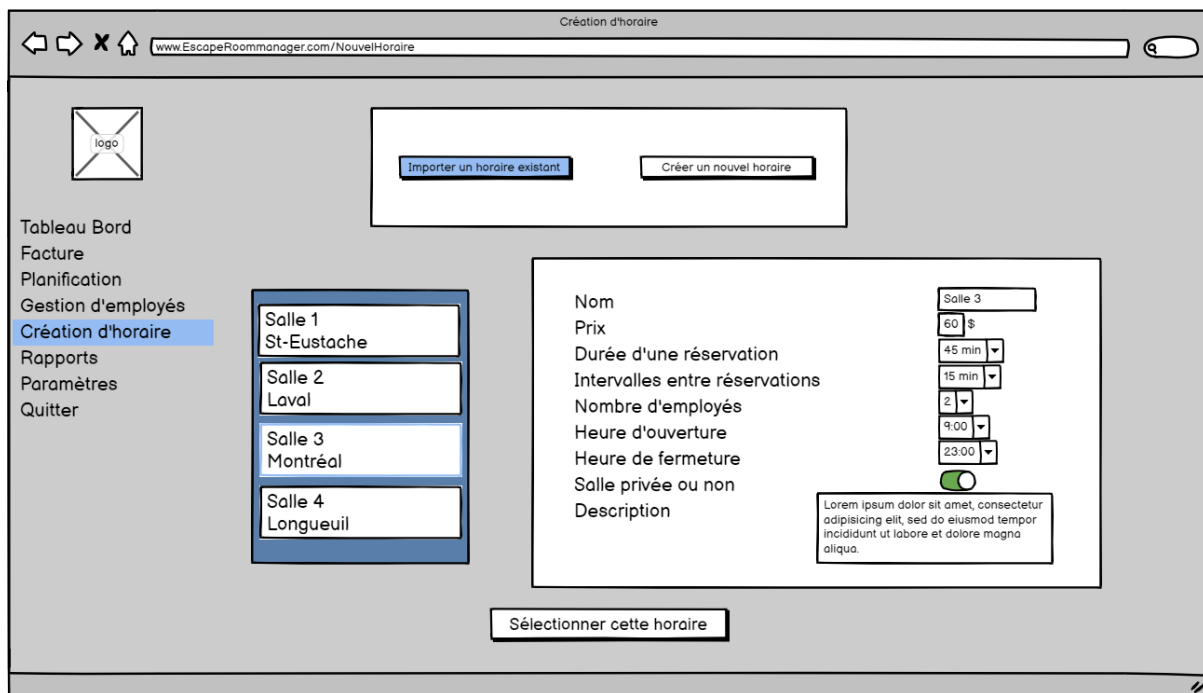


Figure 5 Maquette de la page pour créer un horaire à partir d'un horaire déjà existant

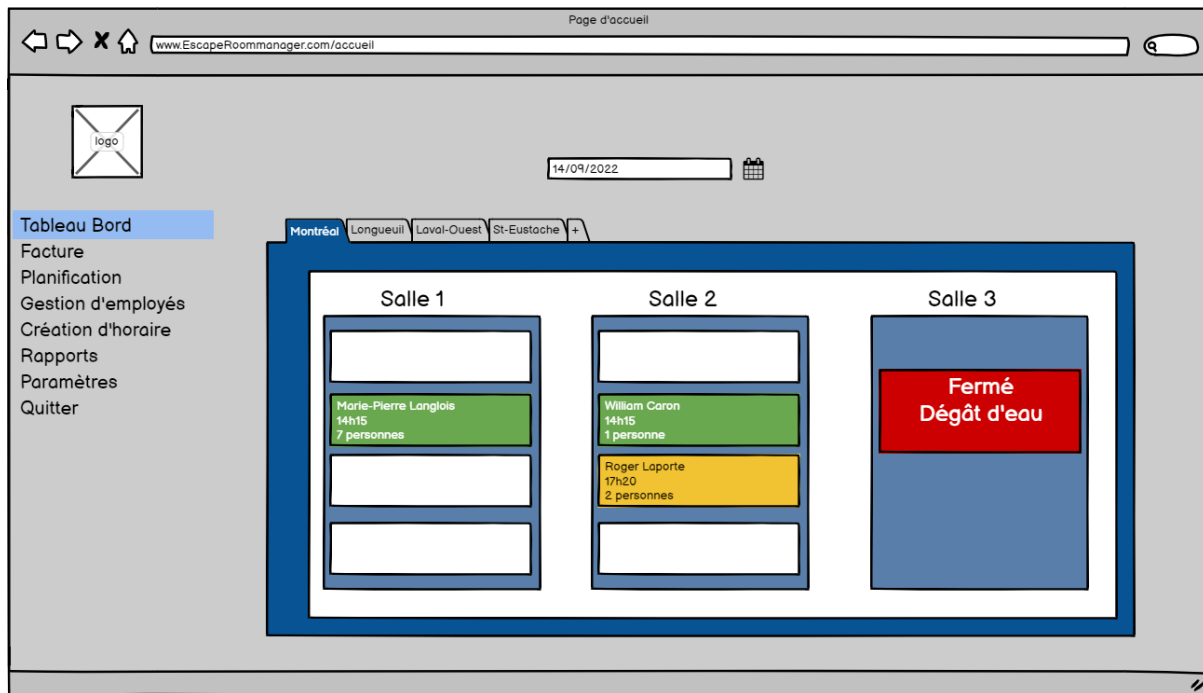


Figure 6 Maquette du tableau de bord — Point de vue de l'employé

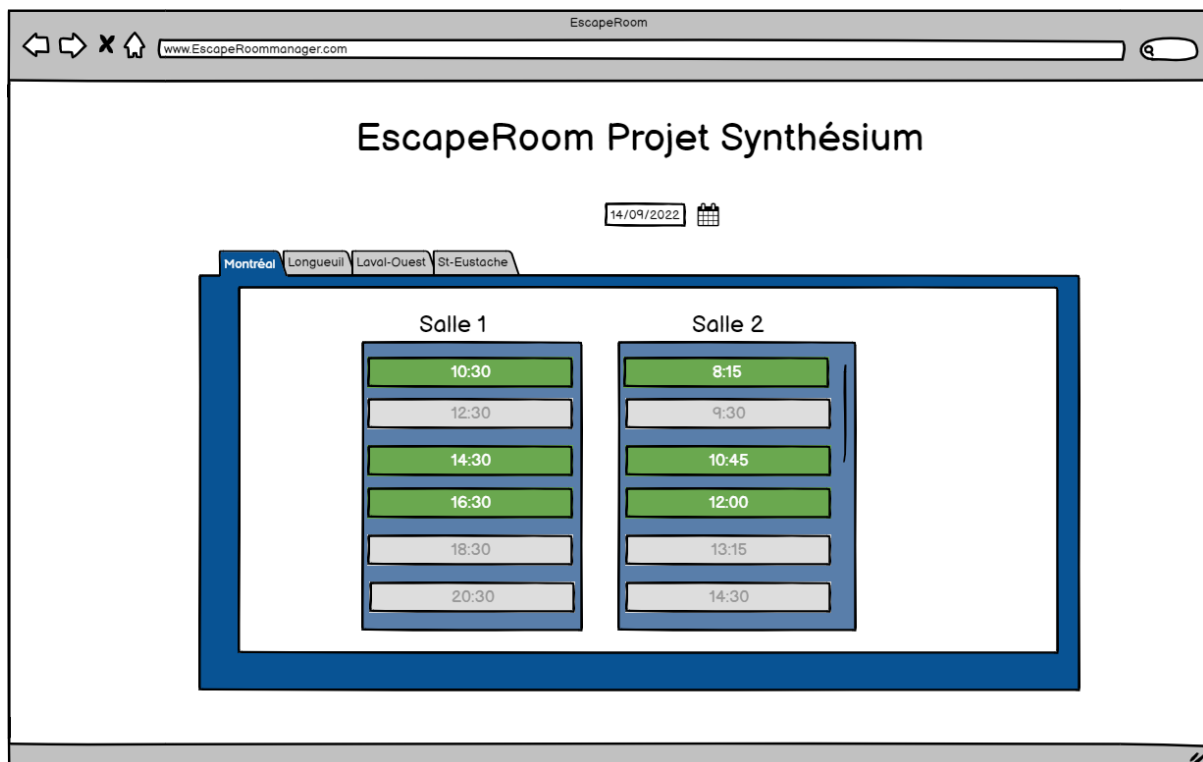



Figure 7 Maquette de l'intégration des horaires sur le site web du client

Point de vue client et employé



Salle 2

Nombre de participants

3

Nom

Prénom


Téléphone

Courriel

Adresse du jeu

Confirmer la réservation

Point de vue employé



Salle 2

Nombre de participants

6

Nom

Prénom

Téléphone

Courriel

Adresse du jeu

Blow

Joe

514-222-333

email@hotmail.com

255 Ontario

Modifier

Supprimer

Figure 8 Maquette pour ajouter ou modifier une réservation




Tableau Bord

Facture

Planification

Gestion d'employés

Création d'horaire

Rapports

Paramètres

Quitter

www.EscapeRoommanager.com/paramètre

Paramètre

Ajouter une succursale

Nom :

Adresse :

Ajouter des salles

Salle 1

Salle 3

Salle 4

Ajouter une succursale

Supprimer ou modifier une succursale

Nom :

Montréal

Adresse :

255 rue Ontario

Ajouter des salles

Salle 1

Salle 2

Salle 3

Modifier la succursale

Supprimer la succursale

Figure 9 Maquette pour ajouter, modifier ou supprimer une succursale

Création d'horaire

www.EscapeRoommanager.com/NouvelHoraire

logo

Tableau Bord
Facture
Planification
Gestion d'employés
Création d'horaire
Rapports
Paramètres
Quitter

Carlens Belony
Admin

Maxence Guindon
Admin

Joe Blow
Employé

Annie Lafontaine
Employée

Nom
Prénom
Téléphone
Courriel
droit accès

Blow
Joe
514-438-0009
courriel@hotmail.com
Emp

Supprimer employé

Enregistrer les modifications

Enregistrer comme nouvel Employé

Figure 10 Maquette pour la gestion des employés qui peuvent se connecter à l'application

Création d'horaire

www.EscapeRoommanager.com/NouvelHoraire

logo

Tableau Bord
Facture
Planification
Gestion d'employés
Création d'horaire
Rapports
Paramètres
Quitter

Carlens Belony
Admin

Maxence Guindon
Admin

Joe Blow
Employé

Annie Lafontaine
Employée

Nom
Prénom
Téléphone
Courriel
droit accès

Supprimer employé

Enregistrer les modifications

Enregistrer comme nouvel Employé

Figure 11 Maquette pour ajouter un employé pouvant se connecter à l'application

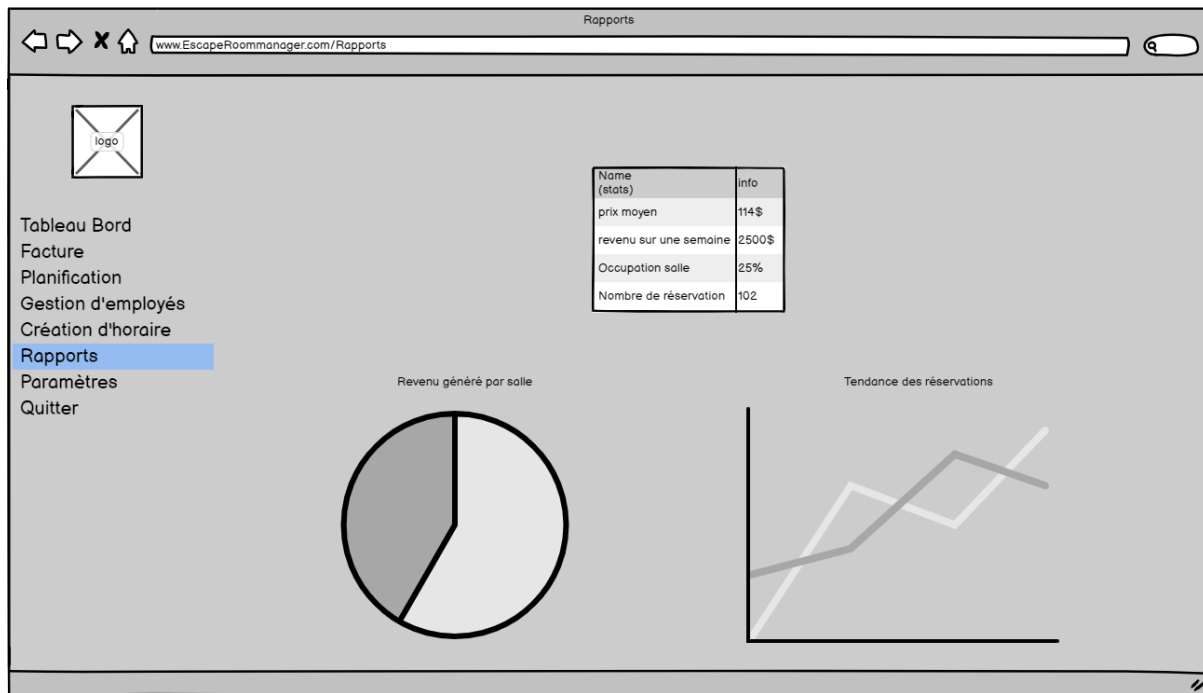


Figure 12 la maquette de la page rapports

Conception UML

Diagrammes des cas d'usages

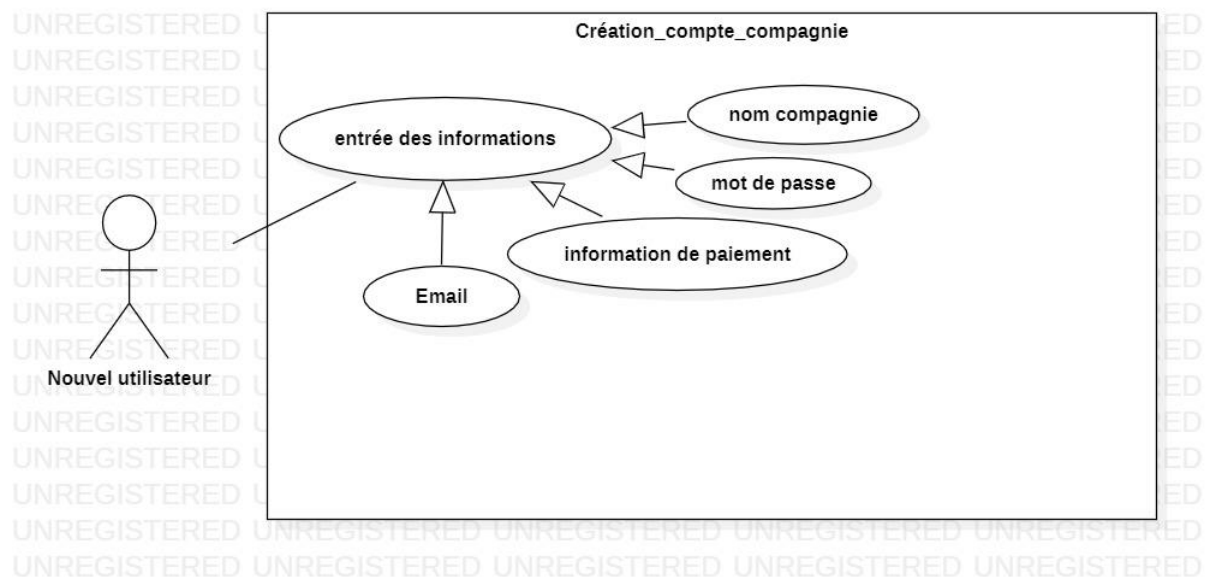


Figure 13 Scénario d'utilisation dans le cas de la création d'un compte

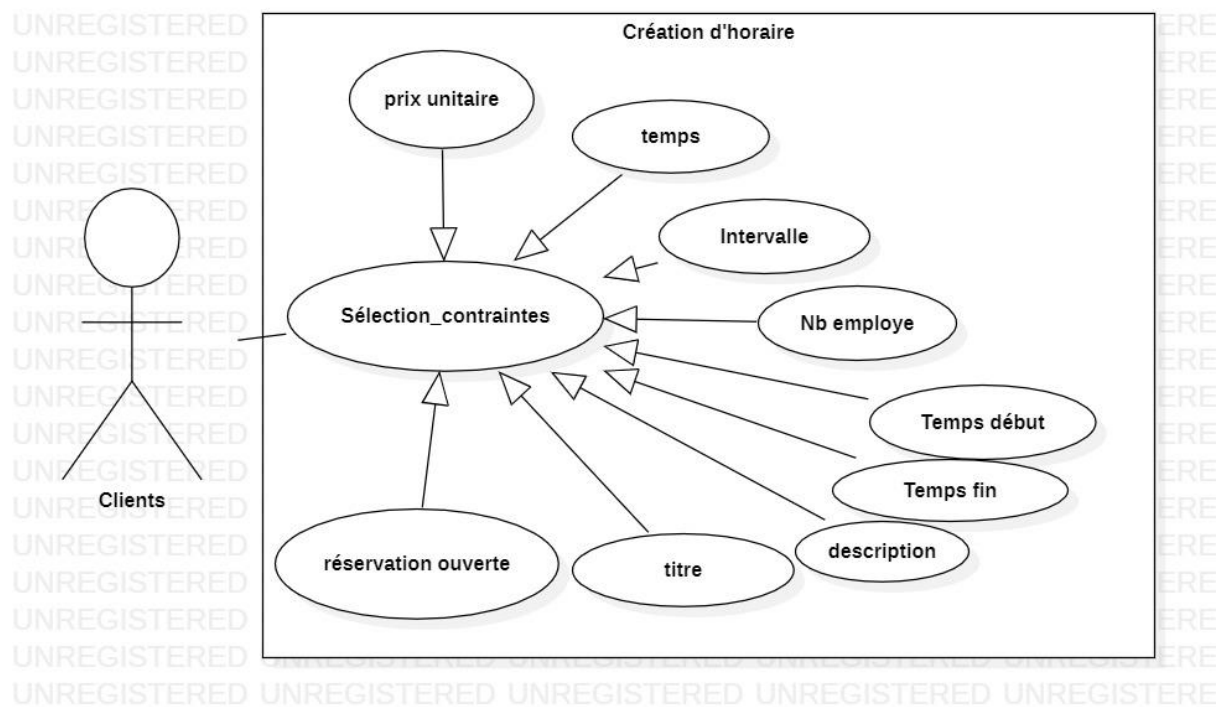


Figure 14 Scénario d'utilisation lors de la création d'horaires

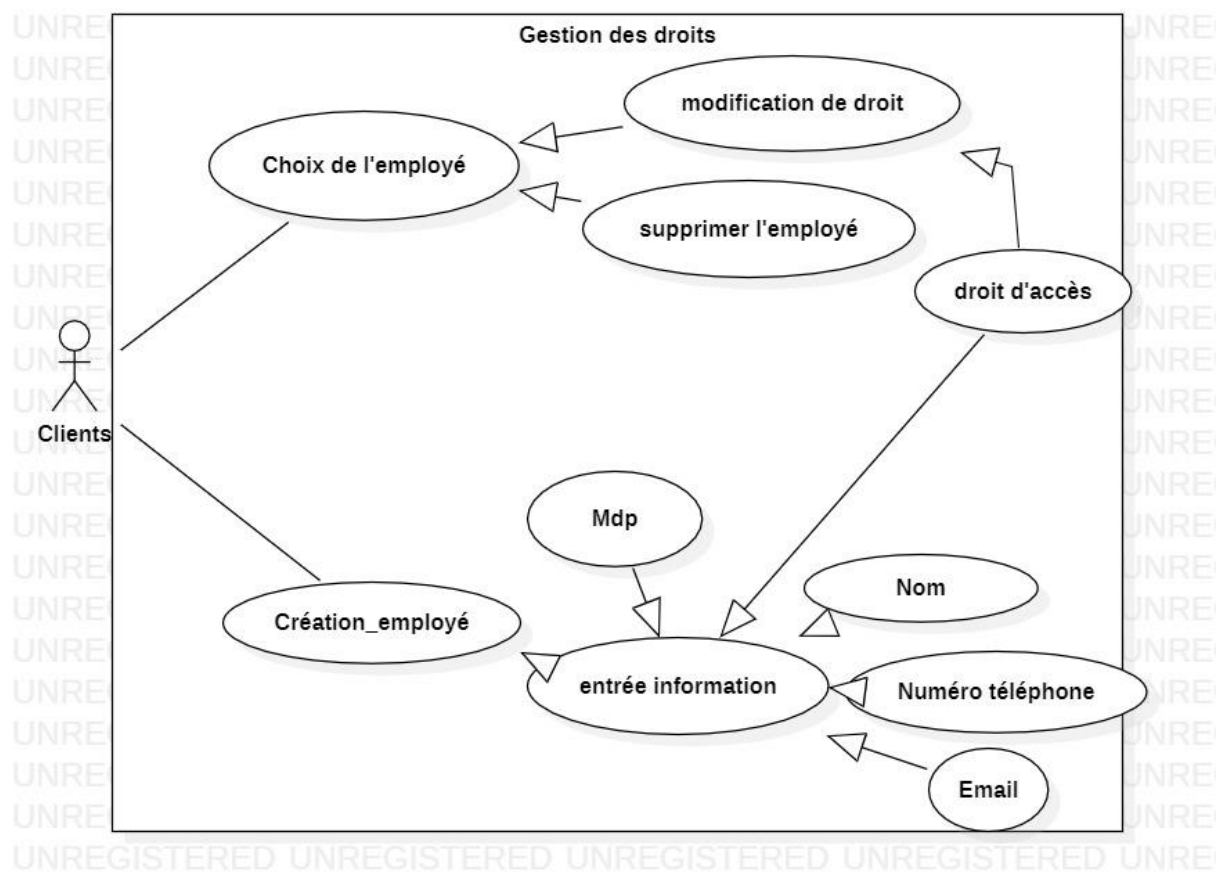


Figure 15 Scénario d'utilisation pour la gestion des droits d'utilisation

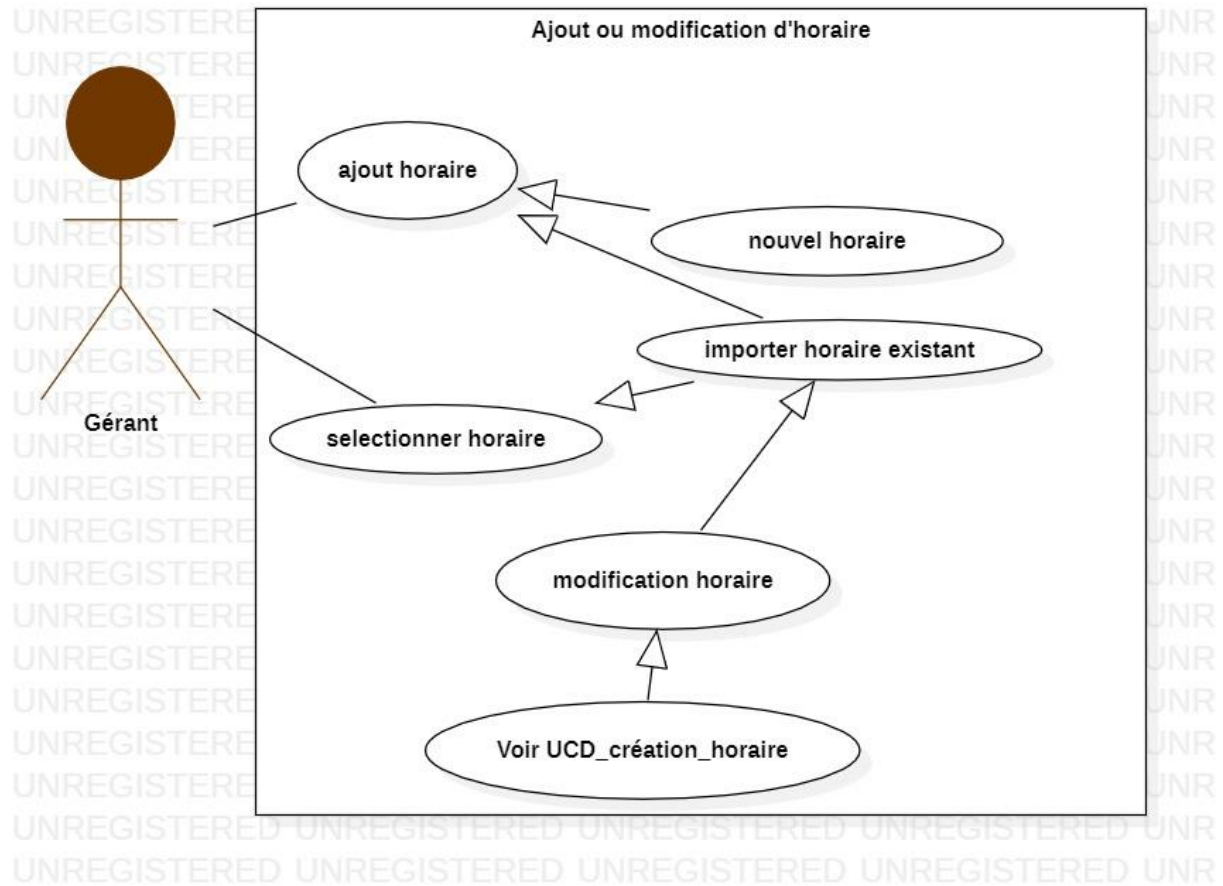


Figure 16 Scénario d'utilisation pour ajouter ou modifier les horaires

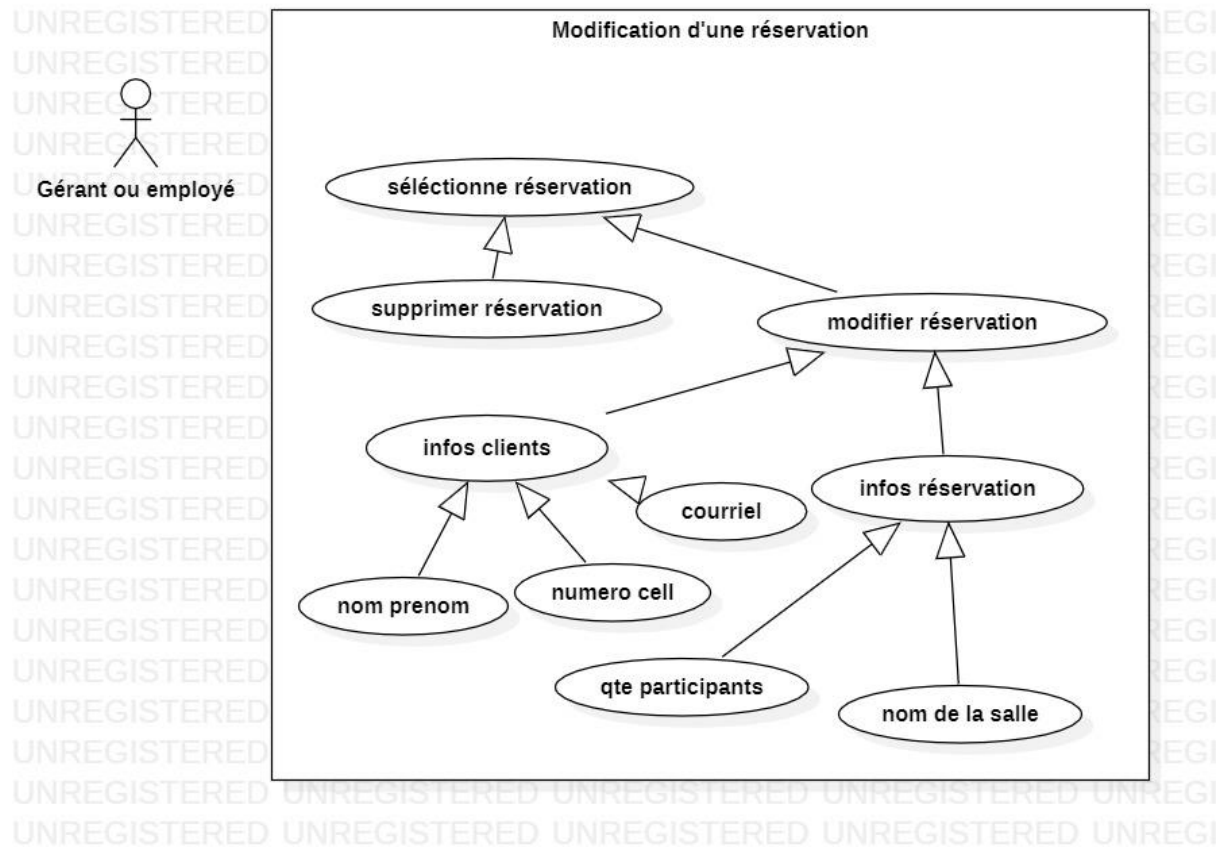


Figure 17 Scénario d'utilisation pour modifier une réservation

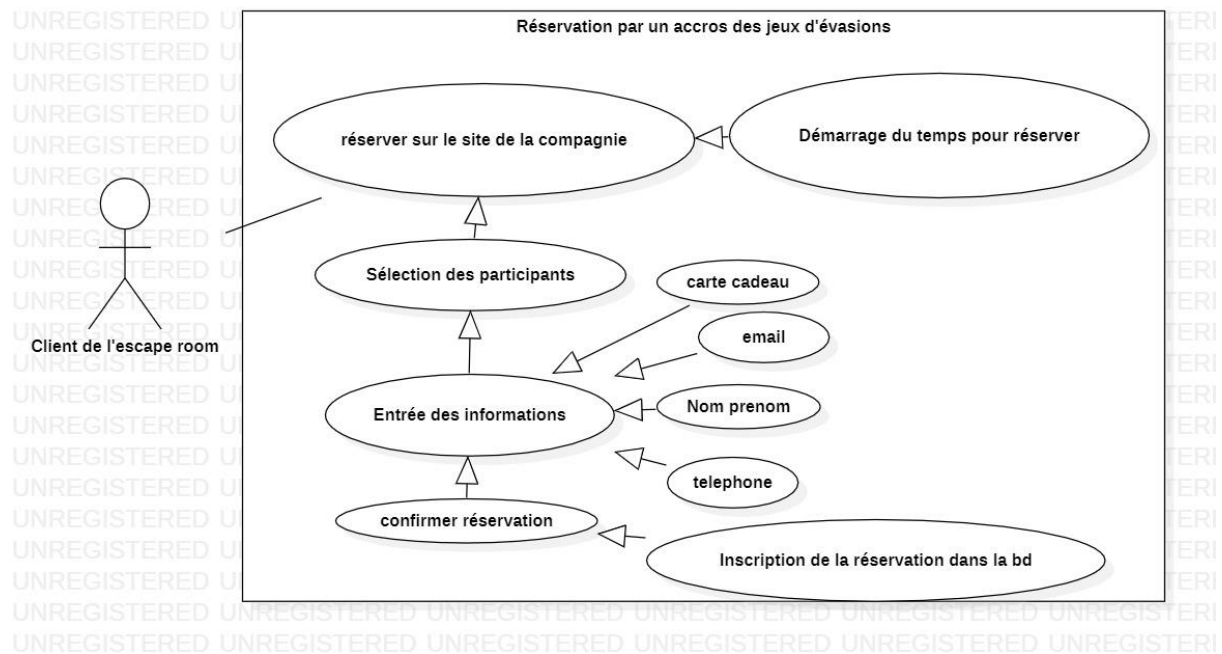


Figure 18 Scénario d'utilisation pour une réservation faite par un client de l'entreprise utilisateurs du service

Diagrammes des classes détaillés

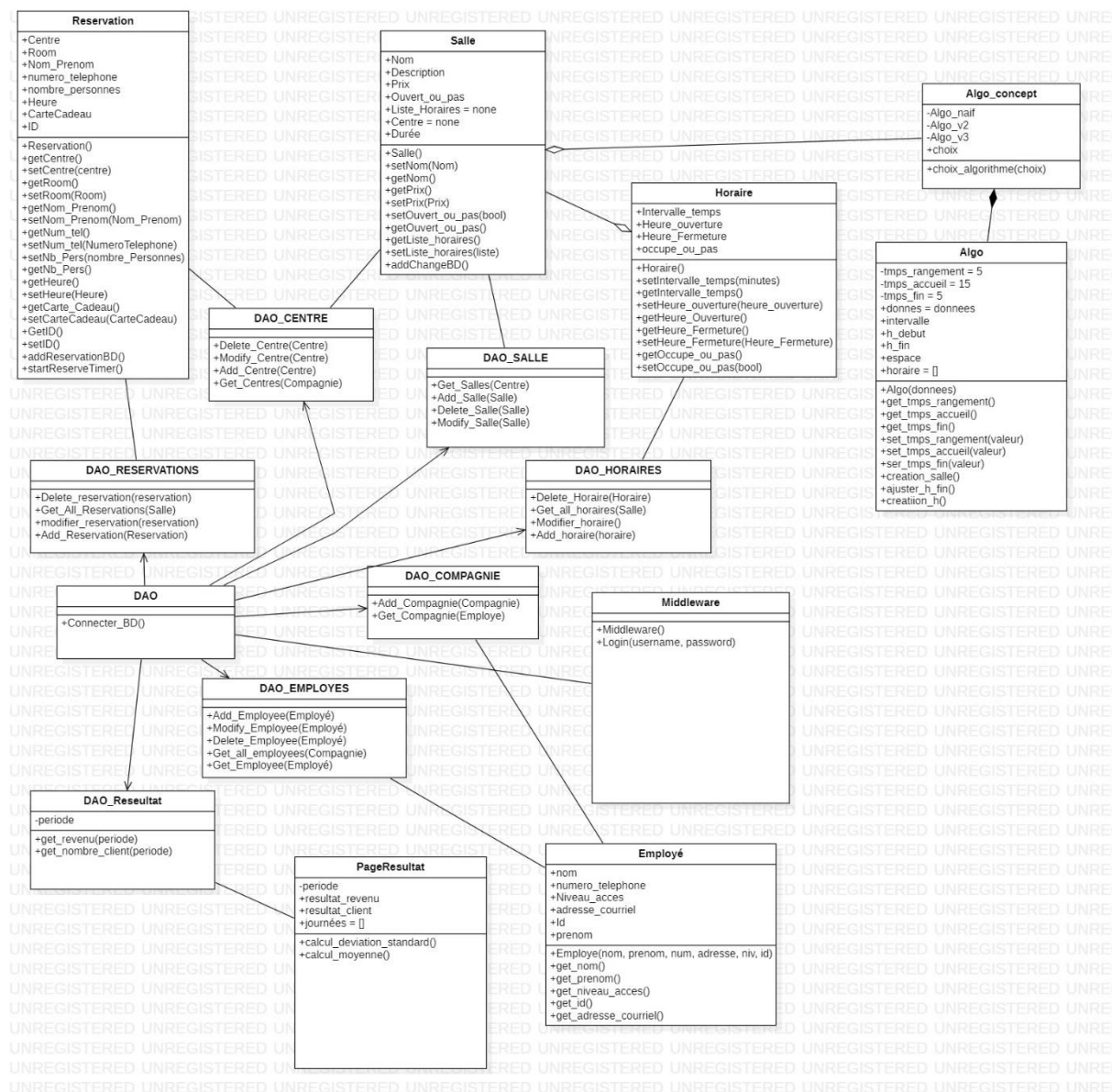


Figure 19 Schéma UML des classes du modèle. La classe algo est un Template pour les futures classes d'algorithmes.

Schéma de la structure de données externes

Pour pouvoir communiquer, notre système devra être capable de communiquer entre deux langages différents. JavaScript et React au niveau de la vue et python au niveau du modèle et du contrôleur.

De plus, nous allons avoir une base de données externes pour la gestion de nos données en SQL.

Ainsi, il faudra utiliser des fichiers en JSON pour être capable de transférer l'information de la vue au contrôleur et vice-versa.

Cette structure en JSON servira essentiellement à transférer les informations des réservations, des salles et de leur horaire à la vue pour qu'elle puisse afficher cette information à nos utilisateurs et

leur clientèle. Elle sera également utile pour récupérer de l'information de ceux-ci par exemple pour enregistrer une nouvelle réservation dans la base de données.

Les informations de transit les plus communes seront celles-ci :

- Les informations reliées à une réservation :
 - La date de la réservation
 - L'activité réservée
 - Le nom du client
 - Le numéro de téléphone
 - Le statut de la réservation (payé ou non)
 - Le nombre de personnes
 - Le courriel
 - L'heure de la réservation
 - Si un coupon rabais a été utilisé
 - Le prix total
 - Les types de clients (adulte, enfant, etc.
- Les informations relatives aux salles
 - Le nom de celle-ci
 - Le centre à laquelle elle est reliée
 - Le nombre de joueurs maximum
 - Si elle est privée ou non
 - Son prix unitaire
 - Sa durée
 - Sa liste d'horaire
- Les informations relatives aux horaires
 - Les heures d'ouvertures et de fermetures de l'entreprise de nos clients
 - La salle concernée
 - L'intervalle entre les départs
- Les informations relatives à la compagnie de nos clients
 - Le nom de celle-ci
 - Ses informations de paiement
 - Ses informations de connexion

Ainsi que d'autres informations importantes comme celles reliées aux succursales possédées par nos clients et leur employé pouvant se connecter à l'application.

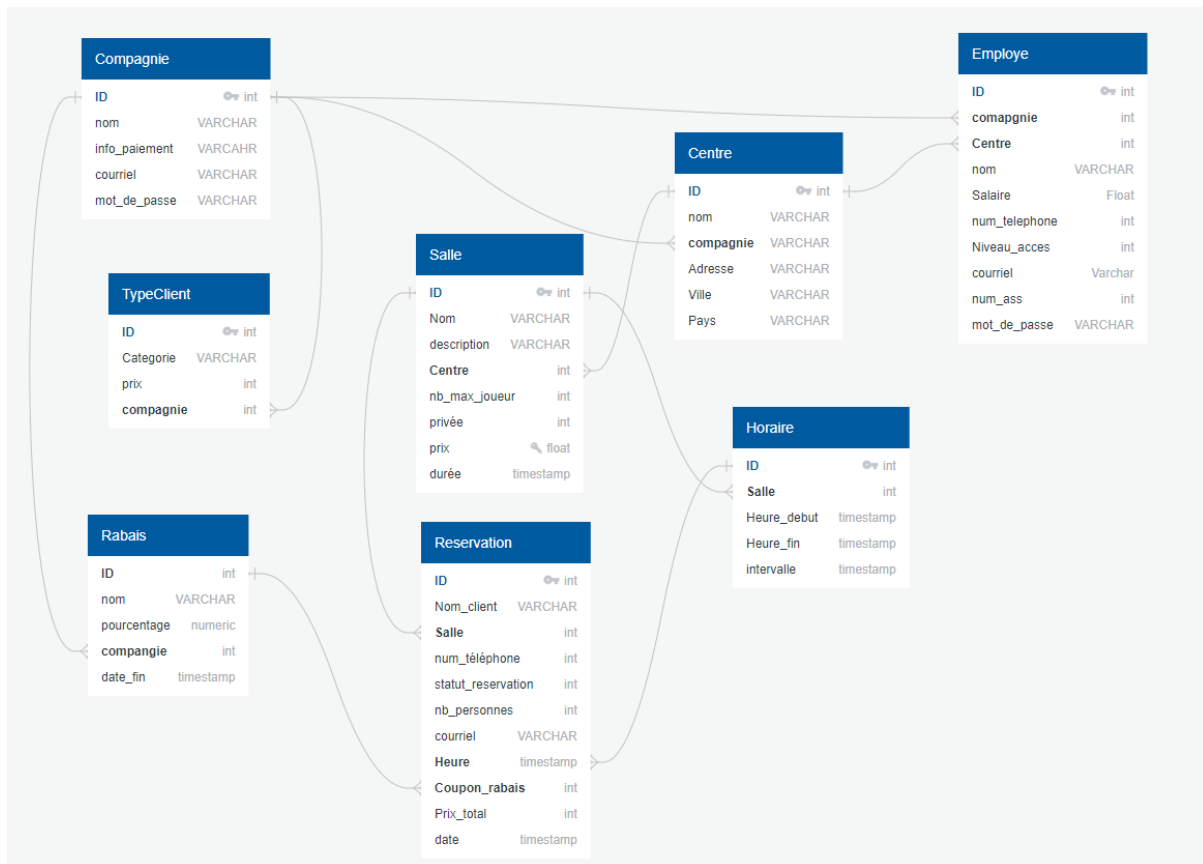


Figure 20 Schéma estimé de la base de données

Au niveau de la base de données, le même type d'information sera nécessaire. Grâce au patron de conception du DAO, nous serons en mesure de conserver nos données sur MySQL.

Au niveau des images, à la suite de la suggestion de Frédérique Thériault, les téléversements d'image pour les salles se feront dans un dossier Upload. Pour être sûre de bien préserver l'image, l'image sera renommée, selon l'ID que nous lui assignerons.

Éléments de conception

Structure de données

Pour notre projet, nous avons déterminé quatre structures de données pertinentes :

- Une liste chaînée pour les réservations
- Un Dictionnaire pour passer des informations à l'algorithme (information pour créer les salles avec leur horaire)
- Un vecteur pour les horaires des salles
- Un *binary tree* pour garder en mémoire les compagnies qui ont souscrit à notre produit.

Tout d'abord, nous ferons l'implémentation de la liste chaînée dans notre modèle. Celle-ci sera pratique pour la gestion dynamique des réservations que notre système prendra en charge. Nous voulons implémenter une liste double chaînée pour être capable d'aller dans les deux directions de la liste facilement puisqu'il sera possible pour un usager de faire défiler celle-ci soit dans le futur ou dans le passé.

L'avantage pour nous d'utiliser une structure de données comme celle-ci, est que lorsque les réservations sont modifiées pour changer de date, le changement dans la structure de donnée sera rapide et élégante puisqu'au lieu de passer par un vecteur qui devrait se cloner en ajoutant la nouvelle donnée et ensuite supprimé son ancienne version de lui-même (tel que nous l'avons vue en classe), la liste chaînée n'aura qu'à changer les pointeurs pour que la réservation se retrouve au bon endroit chronologique dans la liste. Pour la réussir, nous allons suivre différents tutoriels¹ et y apporter les éléments pertinents et inhérents aux projets.

Ensuite, comme nous allons communiquer beaucoup par fichier JSON entre la vue et le modèle, nous aurons également besoin d'utiliser la structure des dictionnaires. Ceux-ci seront pratiques non seulement pour fournir les informations des entreprises utilisant notre produit à la base de données, mais également à la création des objets de notre modèle, comme les salles, les horaires et les réservations. Plus spécifiquement encore, cette structure de données sera des plus pertinentes pour fournir les contraintes à l'algorithme de création d'horaires qui pourra alors les décortiqués rapidement pour créer plus facilement un horaire à proposer pour les salles.

Comme troisième structure de données essentielle, nous allons utiliser les listes de python qui sont essentiellement des vecteurs. Nous utiliserons cette structure de données puisque les listes d'horaires que l'algorithme va créer ne seront pas de taille définie. Ainsi, celles-ci peuvent varier selon les besoins de nos clients et surtout dépendamment de leur heure d'ouverture. De plus, si jamais un client venait à modifier ses heures d'opérations, ils pourraient vouloir modifier l'horaire en conséquence, soit en rajoutant des plages horaires, ou en supprimant des plages horaires.

Finalement, la dernière structure de données que nous utiliserons sera un *binary tree* dans le but de conserver en mémoire les compagnies qui souscriront à notre produit. Ainsi, lorsque l'employé d'une compagnie, ou que le propriétaire de celle-ci tentera de se connecter. Notre modèle pourra vérifier rapidement dans l'arbre que la compagnie existe bel et bien et pourra ensuite retourner les informations pertinentes à l'utilisateur et la vue. À l'inverse, si la compagnie ne se retrouve pas dans

¹ (Bilal, 2021)

l'arbre, on pourra alors envoyer un message d'erreur pour indiquer à l'utilisateur de créer un compte pour son entreprise.

Patron de conception

Middleware

Un middleware est un intermédiaire entre le contrôleur et l'interface qui permet de vérifier si une requête est valide avant de l'envoyer au contrôleur. Par exemple, si un usager essaie de s'identifier, la requête passera par le middleware qui vérifie si le nom d'utilisateur existe (et s'accordent ensemble). S'il n'existe pas, il ne prendra pas la peine d'envoyer les informations de connexions au contrôleur puis au DAO, ce qui prendrait légèrement plus de temps².

Notre middleware nous servira aussi d'API entre le front-end React et le backend en Python Flask.

Au lieu d'avoir de nombreuses fonctions JavaScript, on aura un seul gros fichier Middleware contenant toutes les fonctions requises pour communiquer avec le backend en Python (Flask), et chaque page n'aura qu'à appeler une requête de l'API relié à sa fonction. Par exemple, la page *Login* pourra appeler la requête LOG_IN à l'API du middleware qui lui retournera les informations (ou non, si les informations de connexions sont erronées) de l'utilisateur, tandis que la page « ajouter réservation » pourra appeler la requête « ADD_RESERVATION » à l'API en passant les informations concernant la réservation par format JSON. On utilisera le framework Redux et React-Redux pour implémenter notre middleware.

Le désavantage de ce format est qu'il est assez nouveau pour nous, nous y avons légèrement touché lors du projet MAGIX de C55 où il a fallu qu'on appelle l'API à de nombreuses reprises. L'avantage est qu'une fois que le middleware sera fonctionnel, la manipulation de ce dernier sera extrêmement facile pour le reste du projet, et le lier avec REACT rendra notre site web beaucoup plus rapide.

DAO

Une classe DAO sera utilisée pour que le Contrôleur puisse communiquer entre la base de données et les requêtes de la Vue ; pour insérer des données (par exemple, un nouvel utilisateur), le Contrôleur passera en paramètre un objet contenant les données requis pour l'INSERTION (l'objet Utilisateur contiendra son nom d'utilisateur, son mot de passe, son adresse courriel, etc.) et le DAO s'occupera de décortiquer ses données et les insérer dans la base de données. La classe DAO

² (Wang, 2022)

contiendra des sous-classes s'occupant de leurs objets respectifs. Par exemple, une sous-classe Utilisateurs_DAO s'occupera exclusivement des Insertions et Select reliés aux utilisateurs. Cela permettra de mieux structurer le code et le rendre plus propre.

On utilisera un DAO pour 2 raisons particulières : premièrement, si l'on veut modifier notre type de base de données, on n'aura qu'à modifier le DAO sans changer de code dans le modèle ou contrôleur. Deuxièmement, les *prepared statement* nous permettront d'éviter les SQL injections, ce qui sécurise un peu plus notre base de données.

Statements vs Prepared Statements

La différence entre les *prepared statement* et les *statement* est la suivante : lorsqu'on exécute une *query* avec un *statement*, on le fait ainsi :

```
$expected_data = 1;  
$query = « SELECT * FROM users where id=$expected_data »;
```

Ce qui exécute le code suivant :

```
SELECT * FROM users where id=1
```

Or, si on veut insérer un code malicieux dans la variable comme ceci :

```
$spoiled_data = « 1; DROP TABLE users; »  
$query= « SELECT * FROM users where id=$spoiled_data »;
```

Le code exécuté sera le suivant :

```
SELECT * FROM users where id=1; DROP TABLE users;
```

Et la table *users* sera supprimée. Or, avec un *prepared statement*, le *query* en question est déjà envoyé et stocké dans la base de données.

```
$db->prepare(« SELECT * FROM users where id=? »);
```

Lorsqu'on exécute un *prepared statement* en envoyant une variable, qu'elle soit malicieuse ou pas, la seule requête qui est effectuée est celle du *prepared statement* et rien d'autre. On ne peut donc pas ajouter d'autres *request* du genre DROP TABLE ou d'autres requêtes malicieuses.

Cette stratégie sera utilisée dans notre projet, car il y aura beaucoup d'input qui seront ajoutées dans la base de données, y compris la page de Login qui peut être accédée par tout le monde, y compris ceux qui ne sont pas des employés et donc ont plus de chances d'avoir des intentions malicieuses.

MVC

L'implémentation du patron de conception dit *modèle vue contrôleur* sera fondamentale dans notre projet. Il consiste à séparer des éléments du programme de façon à ordonner la façon dont les données d'un programme sont présentées et manipulées. Il s'agit davantage d'un patron structurel puisque la vue, le modèle et le contrôleur peuvent encapsuler une ou plusieurs classes pour accomplir leur fonction. Dans notre cas, la vue et le modèle sont facilement représentables par les langages différents que nous utiliserons (JavaScript et Python) avec React pour afficher nos données et Flask pour faire le lien avec notre modèle, soit un ensemble de classes Python qui servira à aller récupérer les données au niveau de la base de données et les envoyer à la vue pour qu'elle les présente. Le contrôleur sera plus distinctif puisque ce sera lui qui fera le lien entre les demandes du client qui utilise notre produit, le modèle et la vue³.

Stratégie⁴

Nous utiliserons le patron de conception stratégie pour l'implémentation de notre algorithme. Comme nous voulons créer un algorithme capable de faire plusieurs actions différentes dans le but de créer un horaire. Soit de créer un horaire pour une salle, un horaire pour plusieurs salles, ou encore un horaire ayant comme contrainte un ratio d'employé par salle, utiliser ce patron de conception nous permettra de faire plusieurs algorithmes pour ces situations spécifiques, mais en plus de permettre plus de modularité au niveau des choix que nos clients pourront faire lorsqu'ils voudront créer que l'algorithme leur génère automatiquement un horaire.

Grâce à ce patron de conception, notre algorithme aura la chance de grandir et d'augmenter en puissance avec le temps (dans l'optique où notre produit pourrait être en service pendant de longue année) puisque l'abstraction qui est faite nous permettra de rajouter des algorithmes plus puissants dans le code sans avoir à faire de modification exhaustive de la partie du programme qui gère la création des horaires.

Expression Régulière

Nous allons utiliser les expressions régulières pour formaliser des contraintes sur les mots de passe que nos utilisateurs devront avoir. Ainsi, il faudra que les mots de passe de nos utilisateurs soient ficelés en fonction des bonnes pratiques en ce domaine. Nous allons utiliser les recommandations de la Commission nationale de l'informatique et des libertés⁵ qui indique qu'un bon mot de passe doit contenir au moins 4 éléments différents soit au moins une majuscule et au moins une minuscule,

³ (Kumar, 2018).

⁴ (Shvets, 2018)

⁵ (CNIL, 2017)

ainsi que des chiffres et des caractères spéciaux. De plus, le mot de passe doit être d'une longueur de 12 caractères au minimum.

Algorithme

Notre algorithme sera implémenté dans la classe servant à créer ou proposer automatiquement un horaire à notre client. Comme mentionné plus haut, nous utiliserons le patron de conception stratégie pour être capable d'avoir plusieurs solutions à différentes mises en situation où l'algorithme devra produire un horaire. En deux temps, il devra d'abord définir le meilleur horaire possible selon les contraintes exposées. Il agira sur les contraintes et informations suivantes :

- Le nom de la salle
- La description
- La durée
- Le prix unitaire
- Privée ou non
- L'heure de début
- L'heure de fin
- L'intervalle entre la fin d'une réservation et le début de la suivante
- Départ décalé entre les activités, si oui combien de temps
- Le nombre d'employés par salle

Cette dernière contrainte est la plus pertinente puisqu'elle viendrait combler un besoin particulier des entreprises qui cherche à limiter le nombre d'employés par salle, tout en gardant un service de qualité. L'algorithme devra également être capable de créer plusieurs salles en même temps ainsi que de s'assurer que les horaires proposés soient en symbiose les uns avec les autres.

Tout d'abord, nous aurons un algorithme naïf, qui sera utile dans l'optique où un client utilise notre service pour la première fois et désire créer un horaire avec une seule salle. Cet algorithme naïf sera capable de créer un horaire convenable en utilisant uniquement les contraintes suivantes :

- L'heure de début et l'heure de fin ;
- La durée de l'activité ;
- Ainsi que l'intervalle de temps entre les horaires

Ensuite un deuxième algorithme devra aller encore plus loin en pouvant déterminer un horaire pour plusieurs salles en même temps. Il devra donc pouvoir retourner un horaire qui sera en harmonie

avec les autres salles. Il sera également utilisé lorsqu'un usager voudra rajouter une nouvelle salle à son horaire.

Finalement, un dernier algorithme devra aller encore plus loin en utilisant la contrainte d'employé par salle. Il devra donc déterminer selon les données fournies par l'utilisateur un horaire qui sera optimal pour le ratio demandé par l'employeur. Les contraintes supplémentaires à analyser porteront sur le temps d'accueil d'un nouveau groupe, le temps de rangement d'une activité ainsi que le temps pour finir une activité avec le groupe. Grâce à ces trois dernières contraintes, l'horaire proposé devra pouvoir permettre aux employés de gérer leurs salles en limitant au maximum les conflits entre activités (par exemple une salle qui se termine en même temps qu'une autre devrait démarrer).

Pour nous aider à déterminer le meilleur algorithme possible, nous allons explorer différentes pistes de solutions, notamment les algorithmes génétiques.

Mathématique

« Je me demande vers quelle heure j'attire le plus de clients », ou « Je me demande vers quelle période de l'année ma compagnie devient la plus populaire » est une question que tous les CEO se posent. En calculant la moyenne ainsi que la déviation standard des revenus (ou du nombre de clients), on pourra déterminer en pourcentage vers quels jours de l'année 99%, 95% et 68% de la majorité des bénéfices (ou de l'afflux de clients) se produit.

La déviation standard sera calculée comme ceci :

$$\sigma = \sqrt{\frac{\sum (X - \mu)^2}{N}}$$

Où \sum représente la somme de...,

X représente chaque valeur, et

μ représente la moyenne.

On calculera donc chaque valeur — la moyenne, puis on exposera en 2 le résultat pour chaque valeur puis on fera la somme de tous ces résultats combinés. On le divisera par N qui représente la quantité de valeurs additionnées, puis on fera la racine carrée de tout. Pour voir où se trouvent 99% des données, on prend la moyenne des données \pm (la moyenne * 1). Pour 95%, on prend la moyenne des données \pm (la moyenne * 2). Pour 68%, c'est * 3.

Statistique en lien avec les jeux d'évasion :

- Occupation des salles
- Revenus générés par celle-ci
- Évolutions des réservations
- Prix moyen d'une salle
- Revenu sur une semaine
- Nb de réservation
- Nb de joueur
- Nb de réservation par jour
- Revenu par jour
- autres

Outils technologiques du projet

Pour notre projet nous avons l'intention d'utiliser les technologies React et Flask. Nous utiliserons le cours de veille pour en apprendre plus sur celle-ci et réussir à les intégrer à notre projet. Nous pensons que ces technologies sont pertinentes pour plusieurs raisons. Nous pensons que React nous aidera à présenter un site web moderne et réactif qui permettra à nos utilisateurs de bien gérer leurs salles et activités. L'autre qualité de React c'est que celui-ci est capable d'actualiser seulement les éléments à actualiser d'une page ce qui est non-négligeable pour notre produit qui propose de présenter et gérer en temps réel des réservations d'horaires. Au niveau de Flask, l'avantage de cette technologie est qu'elle performe bien sous pression et il est très efficace pour faire des applications web.

Nous utiliserons aussi Redux et React-Redux. Redux est une librairie JavaScript qui nous permettra de mettre en place notre propre MiddleWare (et ainsi notre propre API) qui nous permettra de faire le lien entre notre application web et notre backend en python. React-Redux nous permettra de convertir les données reçues de nos données en python en données lisibles et utilisables pour React.

Bibliographies

- Aan. (2011, November 24). *StackOverflow*. . Récupéré sur How can prepared statements protect from SQL injection attacks? : <https://stackoverflow.com/questions/8263371/how-can-prepared-statements-protect-from-sql-injection-attacks>
- Academind. (2016, Septembre 25). *ReactJS / Redux Tutorial - #6 Redux Middleware*. Récupéré sur Youtube: <https://www.youtube.com/watch?v=tfuZ7uZmVyg>
- Academind. (2016, Septembre 28). *ReactJS / Redux Tutorial — #7 Connect ReactJS and Redux*. Récupéré sur Youtube: <https://www.youtube.com/watch?v=tfuZ7uZmVyg>
- Bhandari, P. (2020, Septembre 17). *How to Calculate Standard Deviation (Guide) | Formulas & Examples*. Récupéré sur scribbr: <https://www.scribbr.com/statistics/standard-deviation/>
- Bilal, M. (2021, Février 05). *Tuto Python : Liste simplement chaînée : Créer, parcourir, inserer, supprimer*. Récupéré sur cours-gratuit: <https://www.cours-gratuit.com/tutoriel-python/tutoriel-python-les-listes-chaînes-premiere-partie>
- Carlens Belony, M. G. (2022, Septembre 28). *balsamiq*. Récupéré sur Your first project: <https://balsamiq.cloud/spa4zrx/pdckcpz/rD049>
- CNIL. (2017, Janvier 27). *CNIL*. Récupéré sur Les conseils de la CNIL pour un bon mot de passe: <https://www.cnil.fr/fr/les-conseils-de-la-cnil-pour-un-bon-mot-de-passe#:~:text=Un%20bon%20mot%20de%20passe%20doit%20contenir%20au%20moins%2012,est%20%C3%A9quip%20de%20s%C3%A9curit%C3%A9s%20compl%C3%A9mentaires%20!>
- Kharve, D. (2019, Octobre 5). *Medium*. Récupéré sur Creating Middlewares with Python Flask: <https://medium.com/swlh/creating-middlewares-with-python-flask-166bd03f2fd4>
- Kumar, S. (2018, Février 08). *MVC Design Pattern*. Récupéré sur Geeks for Geeks: [https://www.geeksforgeeks.org/mvc-design-pattern/#:~:text=The%20Model%20View%20Controller%20\(MVC,but%20not%20for%20complete%20application](https://www.geeksforgeeks.org/mvc-design-pattern/#:~:text=The%20Model%20View%20Controller%20(MVC,but%20not%20for%20complete%20application)
- Meta. (2022). *Tutorial: Intro to React*. Récupéré sur React: ReactJS / Redux Tutorial - #6 Redux Middleware

Shvets, A. (2018, Décembre 5). *Strategy*. Récupéré sur refactoring guru:

<https://refactoring.guru/design-patterns/strategy>

Wang, X. (2022, Janvier 10). *What is HTTP middleware? Best practices for building, designing and using middleware*. Récupéré sur moesif blog:

[https://www.moesif.com/blog/engineering/middleware/What-Is-HTTP-](https://www.moesif.com/blog/engineering/middleware/What-Is-HTTP-Middleware/#:~:text=Middleware%20is%20a%20design%20pattern,on%20the%20core%20business%20logic)

[Middleware/#:~:text=Middleware%20is%20a%20design%20pattern,on%20the%20core%20business%20logic](https://www.moesif.com/blog/engineering/middleware/What-Is-HTTP-Middleware/#:~:text=Middleware%20is%20a%20design%20pattern,on%20the%20core%20business%20logic).