**🏅 Évaluation**

Perf Conception - TP – application de gestion d’hôtel

*Travail en groupe : Hicham -Tristan - Antoine*

Lien [GitHub](https://github.com/livecampus-projects/blue-moon)

[**I - Analyse des besoins 2**](#_fnwi2z3a8o3t)

[1. Lister les acteurs 2](#_4oy9u2cg8y3d)

[2. Fonctions principales du système 2](#_z1uyqdnwnbpu)

[3. Dictionnaire de donnée 4](#_ixpezdjyfzur)

[**II - Modélisation des données 6**](#_o3uqjidtseit)

[1. Modèle Conceptuel des Données (MCD) 6](#_4af3l8wmvu5v)

[2. Passage du MCD au MLD 7](#_ih3gv0awhes9)

[3. Choix du SGBD 7](#_3p5153zxn59)

[4. Passage du MLD au MPD 8](#_m2emz1miehd1)

[**III - UML 9**](#_b6rjs1u0sxeg)

[1.Les cas d’utilisation (use cases) 9](#_khky2ydy74du)

[2. Diagramme de séquence 11](#_wdecoml0z066)

[3. Diagramme de classe 13](#_4gc2gpgwuryo)

[4. Diagramme d’activité (check-in client) 14](#_4crgadwsskl2)

[**IV - Design Patterns 16**](#_unjohj9filfz)

[1. Patterns utiles pour ce projet 16](#_elka4vl3k92b)

[**V - Documentation 18**](#_6w4j2jjeg8kz)

[1. README 18](#_ksa7bu3yg61g)

[2. Rétroplanning de la conception 18](#_xu3orrty6lr1)

[3. Planning prévisionnel dev 18](#_igyoeuwgym0)

[4. Support de présentation (slides) 19](#_lu1dbbvs3wa4)

# 

# I - Analyse des besoins

## Lister les acteurs

| **Client**  Recherche / résa / reçoit confirmation |
| --- |
| **Réceptionniste**  crée/modifie/annule résa, enregistre paiement, check in/out, consulte historiques clients |
| **Admin**  Gère chambres, tarifs, crée comptes utilisateurs avec rôles (client, réceptionniste, admin), paramètres (saisons, réductions, taxes …) |
| **Gestionnaire des stocks**  Suit stock général, recharge minibar |
| ***Système de paiement*** |
| ***Service d’email*** |

## Fonctions principales du système

* **Gestion des clients**
  + CRUD client,
  + recherche rapide,
  + historique de séjours,
  + calcul auto de l’éligibilité à la réduction 10e réservation,
  + consentements RGPD.
* **Gestion des chambres**
  + CRUD chambre et type,
  + caractéristiques,
  + statut (libre/occupée/maintenance),
  + rattachement minibar.
* **Gestion des stocks minibar**
  + Catalogue produits + catégories,
  + stock général hôtel,
  + stock par chambre,
  + mouvements (entrée, sortie, réassort),
  + seuils d’alerte.
* **Réservations**
  + Création/modification/annulation,
  + moteur de disponibilité par période,
  + attribution de chambre,
  + application réductions et tarifs saisonniers,
  + check-in/out.
* **Paiements**
  + Enregistrement paiements (espèces/carte/en ligne),
  + multi-règlements,
  + remboursements/avoirs,
  + rapprochement avec réservation.
* **Facturation**
  + Génération facture PDF avec TVA et détails consommations minibar,
  + envoi au client,
  + Liste des factures payées.
* **Utilisateurs et rôles**
  + Comptes,
  + rôles (admin/réceptionniste/client),
  + Authentification,
  + journal d’audit.
* **Notifications**
  + Emails de confirmation/modification/annulation,
  + rappels séjour,
  + alertes stock bas minibar.
* **Reporting & tableaux de bord**
  + Taux d’occupation,
  + CA par période,
  + suivi des consommations minibar,
  + exports.
* **Paramétrage**
  + Tarifs par saison,
  + taxes, modèles d’email,
  + moyens de paiement,
  + politiques d’annulation.

## Dictionnaire de donnée

**Table : Clients**

* Utilisateur identifiant → id\_client → INT (PK, AI)
* Nom du client → nom → texte
* Prénom du client → prénom → texte
* Adresse email du client → email → texte (UNIQUE)
* Numéro de téléphone → téléphone → texte
* Adresse postale → adresse → texte

**Table : Statut\_Chambre**

* Identifiant statut ⇒ id\_statut ⇒ entier (PK)
* Libellé statut ⇒ libelle ⇒ varchar(50)
* Couleur statut ⇒ couleur⇒ varchar(7)

**Table : Chambres**

* Identifiant unique de la chambre → id\_chambre → entier (PK)
* Numéro de la chambre → numéro → unique
* Type (simple, double, suite) → type → texte
* Prix par nuit → prix\_nuit → décimal
* Statut → statut → FK (**Statut\_Chambre)**

**Table : Réservation**

* Identifiant unique de la réservation → id\_reservation → entier (PK)
* Référence vers le client → id\_client → entier (FK)
* Référence vers la chambre → id\_chambre → entier (FK)
* Date d’arrivée → date\_arrivée → date
* Date de départ → date\_depart → date
* Montant total calculé → montant\_total → décimal
* Statut (en attente, confirmée, annulée) → statut → texte

**Table : Paiement**

* Identifiant unique du paiement → id\_paiement → entier (PK)
* Référence vers la réservation → id\_reservation → entier (FK)
* Espèces, carte, en ligne → mode\_paiement → texte
* Montant payé → montant → décimal (8,2)
* Date du paiement → date\_paiement → date

**Table : Statut\_Facture**

* Identifiant statut ⇒ id\_statut ⇒ entier (PK)
* Libellé statut ⇒ libelle ⇒ varchar(50)
* Couleur statut ⇒ couleur⇒ varchar(7)

**Table : Facture**

* Identifiant unique de la facture → id\_facture → entier (PK)
* Référence vers la réservation → id\_reservation → entier (FK)
* Date d’émission → date\_facture → date
* Montant facturé → montant → décimal
* TVA → tva → DECIMAL(8,2)
* Statut de la facture → statut → FK (Statut\_Facture**)**

**Table : Produits**

* Identifiant du produit → id\_produit→ INT (PK, AI)
* Nom du produit → nom → VARCHAR(50)
* Categorie → categorie → VARCHAR(50)
* Prix unitaire → prix\_unitaire → FLOAT
* Stock général → stock\_général → INT
* Stock initial minibar → stock\_minibar → INT

**Table : Minibar**

* Identifiant du minibar → id\_minibar → INT (PK, AI)
* Identifiant de la chambre→ id\_chambre → INT (FK)

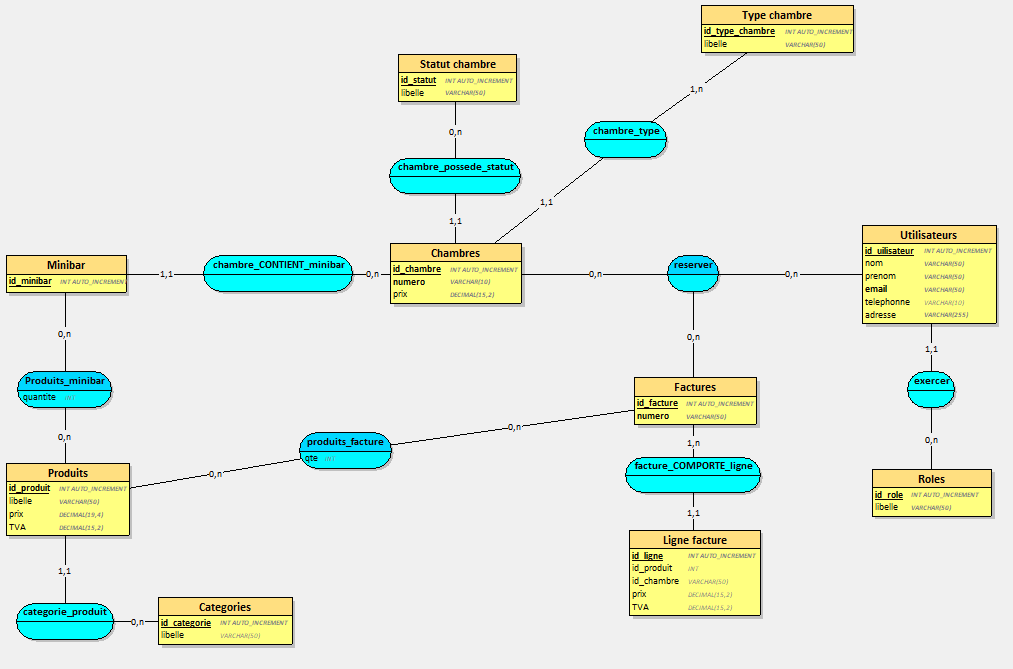
**Table : Produits\_minibar**

* Identifiant du produit→ id\_produit → INT (FK, AI)
* Identifiant du minibar → id\_minibar → INT (FK, AI)
* Quantite

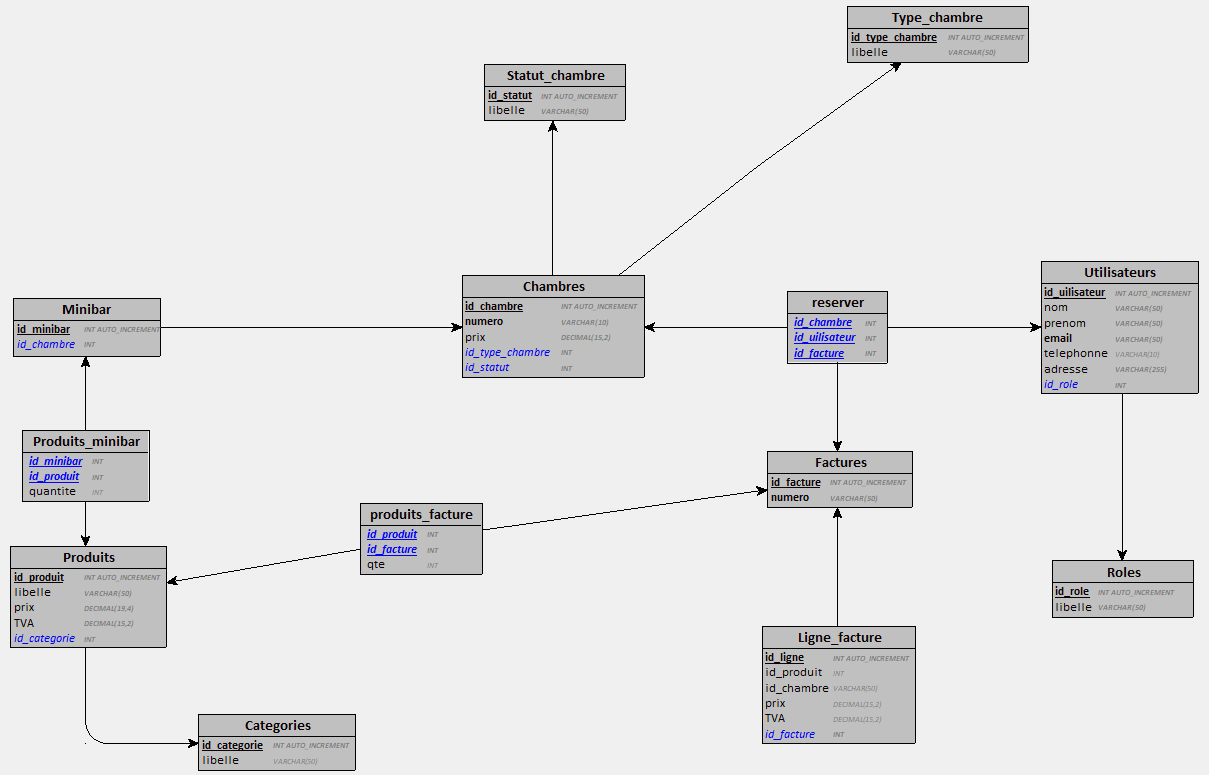
# 

# II - Modélisation des données

## Modèle Conceptuel des Données (MCD)



## 2. Passage du MCD au MLD



## 3. Choix du SGBD

Nous avons choisi **PostGresql** pour les raisons suivantes :

* **Très sécurisé et fiable pour paiements et factures**Le chiffrement côté transport (SSL), le contrôle fin des droits utilisateur et la réplication/sauvegarde avancée. Ça protège les données sensibles et assure une restauration fiable.
* **Rapide pour les gros volumes et bon pour les rapports**

Parce qu’il sait organiser les données de façon très efficace. Il crée des chemins rapides pour chercher l’info (index), il peut préparer à l’avance des tableaux de résultats, et il peut utiliser plusieurs cœurs du serveur pour faire les calculs plus vite. Résultat : même avec beaucoup de lignes, les recherches et rapports restent rapides.

* **Empêche facilement les doublons et surbooking**

Parce qu’on peut dire directement à PostgreSQL de refuser deux réservations qui se chevauchent pour la même chambre. La base vérifie toute seule les dates et bloque automatiquement les doublons ou les chevauchements, sans que tu aies à programmer ça dans ton application.

* **Installation rapide et simple de PostgreSQL**Télécharge PostgreSQL depuis [postgresql.org/download](https://www.postgresql.org/download).  
  Choisir son système (Windows, macOS ou Linux).  
  Installer avec l’installateur (suivant → suivant → mot de passe admin).  
  Lancer pgAdmin (fourni) ou utiliser psql en ligne de commande pour se connecter.  
  Créer la base avec CREATE DATABASE hotel.
* **Gère très bien les relations et contraintes**Parce que PostgreSQL sait lier correctement les tables entre elles (par exemple une réservation reliée à un client). Il vérifie automatiquement que les données respectent les règles que tu as fixées (pas de réservation sans client, pas d’email en double). Ça évite les erreurs et garde les infos propres.

## 4. Passage du MLD au MPD

**Roles** = (id\_role INT AUTO\_INCREMENT, libelle VARCHAR(50) );

**Statut\_chambre** = (id\_statut INT AUTO\_INCREMENT, libelle VARCHAR(50) );

**Factures** = (id\_facture INT AUTO\_INCREMENT, numero VARCHAR(50) );

**Ligne\_facture** = (id\_ligne INT AUTO\_INCREMENT, id\_produit INT, id\_chambre VARCHAR(50) , prix DECIMAL(15,2) , TVA DECIMAL(15,2) , #id\_facture);

**Type\_chambre** = (id\_type\_chambre INT AUTO\_INCREMENT, libelle VARCHAR(50) );

**Categories** = (id\_categorie INT AUTO\_INCREMENT, libelle VARCHAR(50) );

**Chambres** = (id\_chambre INT AUTO\_INCREMENT, numero VARCHAR(10) , prix DECIMAL(15,2) , #id\_type\_chambre, #id\_statut);

**Utilisateurs** = (id\_uilisateur INT AUTO\_INCREMENT, nom VARCHAR(50) , prenom VARCHAR(50) , email VARCHAR(50) , telephonne VARCHAR(10) , adresse VARCHAR(255) , #id\_role);

**Minibar** = (id\_minibar INT AUTO\_INCREMENT, #id\_chambre);

**Produits** = (id\_produit INT AUTO\_INCREMENT, libelle VARCHAR(50) , prix DECIMAL(19,4), TVA DECIMAL(15,2) , #id\_categorie);

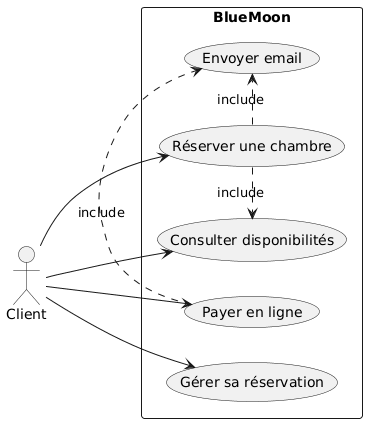
**Produits\_minibar** = (#id\_minibar, #id\_produit, quantite INT);

**reserver** = (#id\_chambre, #id\_uilisateur, #id\_facture);

**produits\_facture** = (#id\_produit, #id\_facture, qte INT);

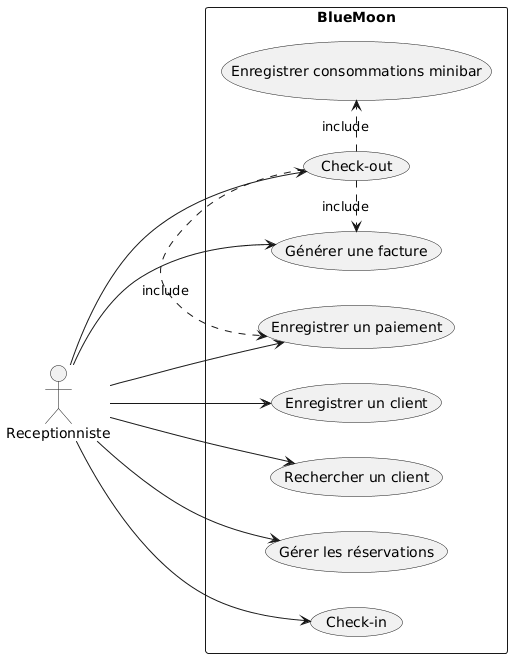
# III - UML

## 1.Les cas d’utilisation (use cases)



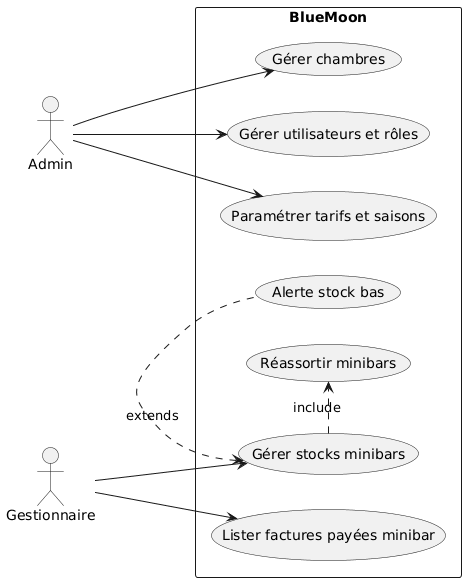
Lien PlantUML : [LIEN](https://editor.plantuml.com/uml/PT7DJiCm30VmUv_2seaTnWDmG1DbuZG93N56NkjrbbndoiSaX7YWFSTUZBYj86Kv_lrpuYJh4D77r4ibT8WG7NZkZX5QzjH4TbfX4vs7MfWqLXQYTaAmU994M-Tq0LyLm5tjD2I9v7DhE3dbFGl7woKi0GEyrn_F5flSvP3yESka1CqH-xsd9TWQvOu26dovNdnM0S6FBMW3bVADvB15nOxCl-9dvgGWtEcq8nGwrurjzEm6s2DBgIQsiJp3QHAuV_mT7-01M1j9BTs0eUq6v3FdzG9yL-EhmchrTyI_R7YBULHSUbwm-wr9s_oRFm00)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



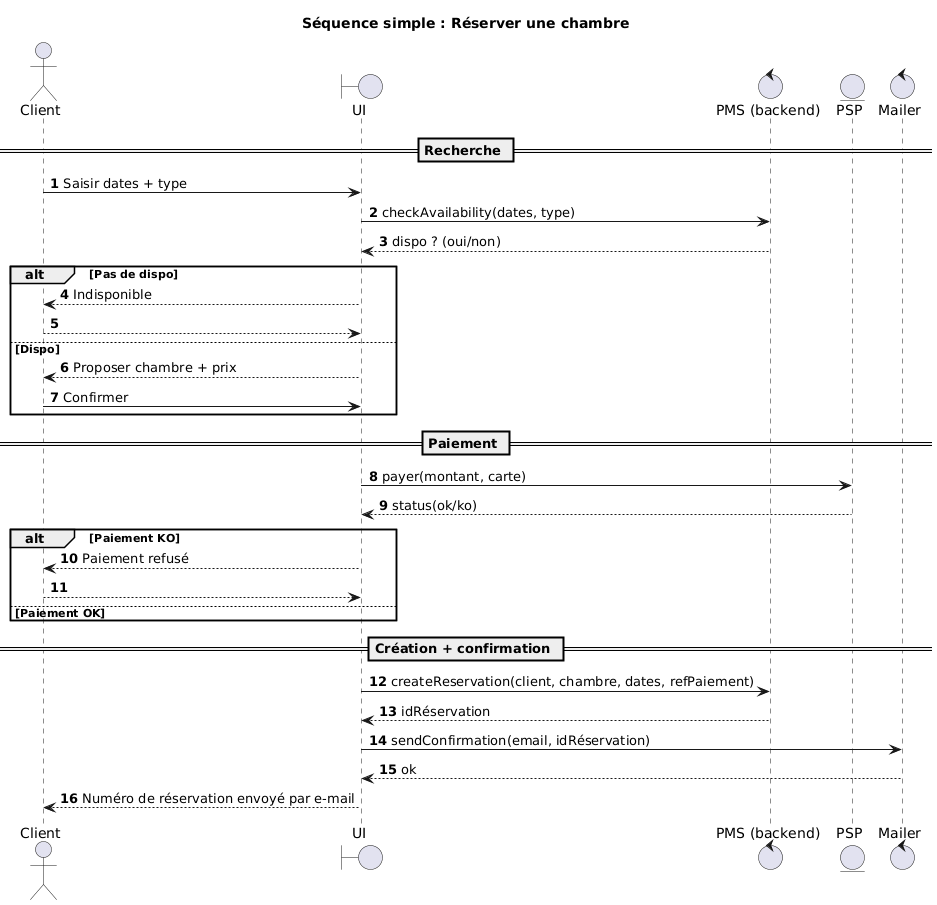
Lien PlantUML : [LIEN](https://editor.plantuml.com/uml/XPBDJWCX58NtynGtNJcByG5SDDM9cYuQJPEk3JAtKo9S6dwQ6EC3zJdwOaAPiRGIIy82piVXNWupwvXnNibAujg1qs14lt7G2OFS2KqLuquRM2B7RLoJi0whA3Bg9SBaGNfSQ4qJ-Ae0RXx9O1yOWmOy0PS2oTKG1hEmQj_kkwuzxaKuk6xGn7d13l2Je8n-FkoZgqGBvh2tQ7OiLcJhNtw9bjc8jXla7usWP9QF11xrEPrGxTqVDaDVl2iqjsK2LIeviQ_iKvsAfLIl9uHrU4LliCwDvxJJWkEbBmyTQQLIQw04YNTcwk7C8YnNDfx9AeFRQQR17GZYqdTbQhZr2ZKqab7VrVaNWAQPdkTPrhC8oq3AhAoDCVsZncJAyXX6MHrVVuRKnU__0m00)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



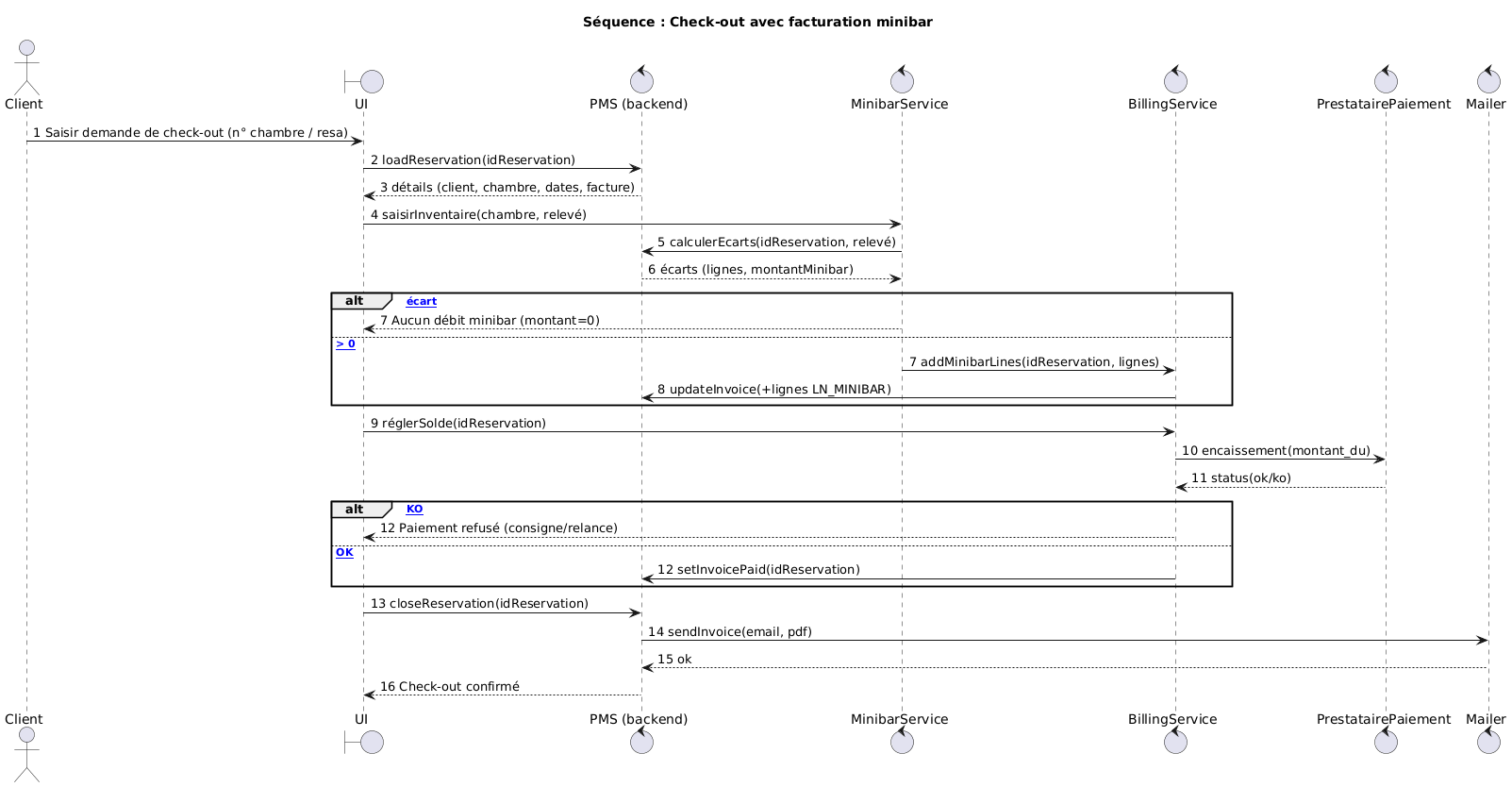
Lien PlantUML : [LIEN](https://editor.plantuml.com/uml/RP9DRi8m48NtESM8LLd00QeAyRD0bK12L5rN9XbWLCUkPYRzKTK3PTyRv6ArSPE6WbUsvtrwRyQUYXhMghIPnRs2Uc0w71KAOioLlCjChfvXLfJKxfSeiU9Cq6HHPjp18WpcjiArzsu0dnd0pRAf6HdoeobtZ3A4YsK47XTFM-zBwG6LaYKnYXKBe08ttxR5VudvR161ZM5JDhL6AhH1-vDU38bt2KZw3LDExj3p4FNviq3eYNQ6px8bv24A8h1jQYFYMOclohjCc6IHcLbanMG2EtFPUa9M_gspMP5eoBKF4wx2iE35V3GrTWc7BRDEvtltwYd7EBOsB8md_Hnm2-HoMnNOrzoD-yP1W--Ahf3iApkzC8n6axytER-AGp-xQSVQ_mzjBJLshVA_WsdmZ__l1m00)

## Diagramme de séquence



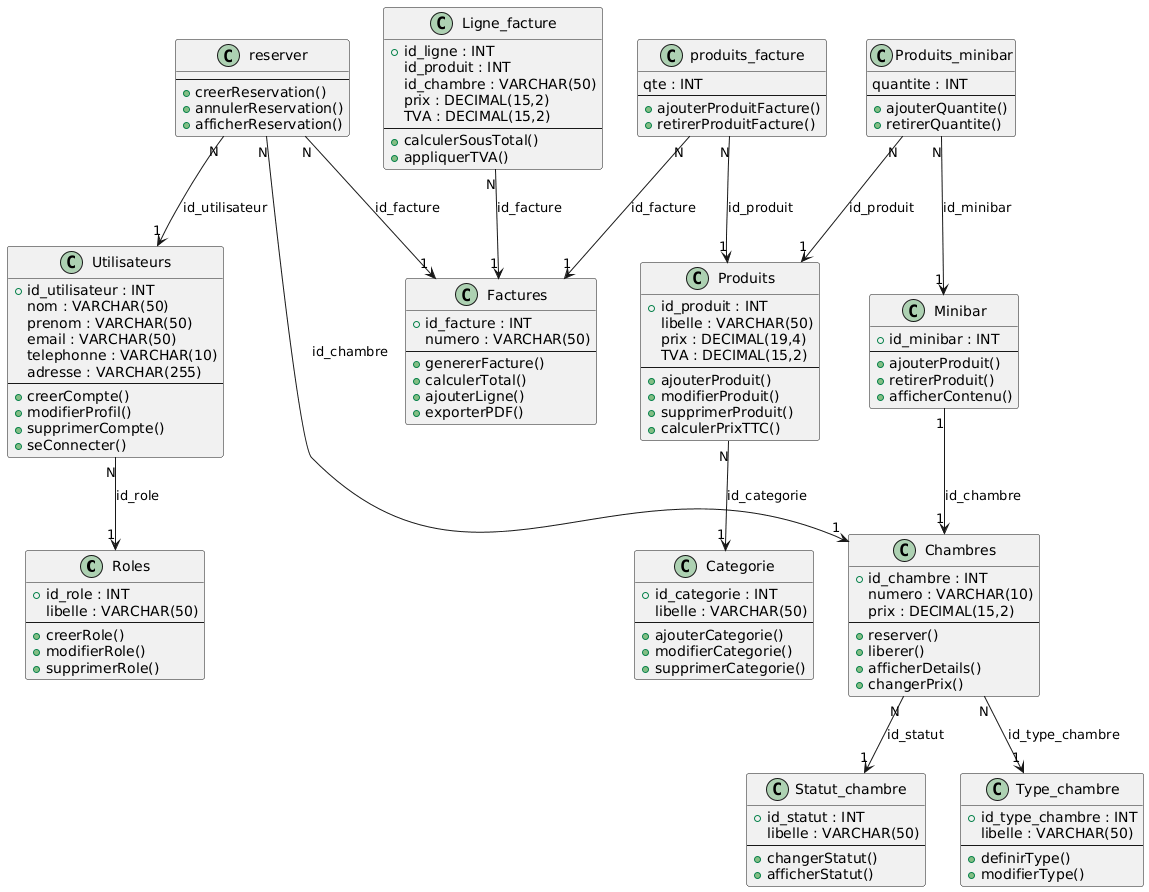
### 

Lien PlantUML : [LIEN](https://www.plantuml.com/plantuml/umla/VPL1Rkis48RtFCKW6oUBDEss28muHXeKg073Hh9fYuAcnlBK5AcIL1AZAD1LBz1tWBTyFeTkafCyaZ9Dcq3Ume0q_ySPpV0dVMSTDwvh97Fa9CAzx72cjOBvR0dl__uFKswX6dME95dko2kNjTGhBg-OnBK3fy5GlN5GaK4H06Qtf5fkU0DSE6sMRYSH-2jQtU29s7AnvJKESbZBLIsHiRW8BgQIKBaByFM7noGi-ht0DbHIP1r6Oe4YoPEgeI0Oxh0pKOsXf3-YZMiv6GJhjDZQ22t3uwmsYLlkhEk_D0WLGii9c_Gry-NyoA1v8O60rmqd6TMPVq33sB4Xk3YTQMH2QcpWRmPmkHpnpcrGELeJcgiWFqt_-4NNf5ZGfwRVem4HHv3agK7kS1XBW7px6pJXLu0_aQeEME0cV-hEeRdfznMv06zuipAOw0SVNMZT1FHPau6ARAiLhKYIw_VsM7w3OZjvuIHPhEubFmWqq2aiSzvhlGtGJ5UnGp037RrqMZY0fEhucKfryWDqofL0cU0u8Iv5n5j3RmcR6n8OSxMjfB-w8LlLHOV2cgf-7tlA0zznwNOnuOCoM0S7eJdk_34ltmNaiT-hVX-0DHUko-t-h5vqg1pp-ESTceCxWe-4LciopLb3IrJLGm1O6dlAQNtrdTzz2y7HAsxoHr1rA6Jpgi7B9VjC-1gj7JS1dJC8_MVmHlXenLDRSOTnJJaDePNL9J_ph_VSJBtMDUmVnaOmiLOBYcrQu28SH0ll_ttoE1li2DVNsUSvb0oJ8uKxJeNI2miKGIZEHmhdqv23XPbpCEHChyKNv6Wkd-F98-ajEo95oe6nU3kTpID7yfdCmLENfKjhYEVDBeMyfwLKcCbVRYTTnCiiW8LHHx10UTXUKa9s5TxWc_C0EmmFlXkV3nT-XDlR0piU5rWyhGMH3oMKq51gFFvD_Vh3ztXsJVpE0idZcZoeRnNFVPLKNXx0OjyA-Cud2V-YNm40)



Lien PlantUML : [LIEN](https://www.plantuml.com/plantuml/uml/XPHBRjim48RtEiN0tIY8ZJpIfAs16daq2oEF616w2eg09iSfOOfC-J3Q8_GesdNH4rGNwuYc6ATUD0XWoVhddty-ZdpgFBC-rAhmqYk4gcs-1TGSOGmNNv4lHoPuO2lai63S1ykyD1fggUMSsQAWxuo52oLH-s9kWXRC_e3Fqu8RxQrHCBkfW3aOT9_bdF4bQh4po8zlraPHaguhj2l9yKLpBfMI-YbgqlMMPbRDNZoPL6WBYlCT1bEzecXCMkn33-3Df_XNG0eEemb57iC1L4mwQK5WpRH0e7-U8PJwpo-wPVMSlFR0ecCx1L1XLqxZZU4Gb67Y7XsbYvnAkNb7weXZbDeTWMWRYgOSb3m66VR-Gn3Cenicw9WRzS36y1PS3FioO9cBBIfSjGrLzVgSyHWuKpmGe4lEh7Ul8-RId7MpvGcqJIo2KiadtUMhYJdJFabs2jG2Y3ijXlBmS1PuqAdeosQOHE0Th1N4OIvzVq1G9jUF-sIe7C93-tFTTmBxhtmc_N9qNao8zE1QQjoQR1sP1kjhCfFt49uxteJII8xbxbeArxUFDzFRwVdPVHmiRTIbvWJUOKtOyoxbyybvFe1jcoVYN1aby7MOVqDKCrg-VU3ikJliVln74KZOFONHflF10JZFV72bMUujJQU9j6TCnb1mTTT1scZHx_SXP8t5HN1j0oKtsdNZxbbKJFEuQcleMNjt5Vssk96VGv-eaLfiJqdO-fr8-2vK-xkR4dO1QoRL5iBaVGHS6OV_VvicwOMd6dedg5b_ZD5-2CzYGTeaWb5kS0nccJmoe9ED7pq2iv2sRfiym2bTX5hz1G00)

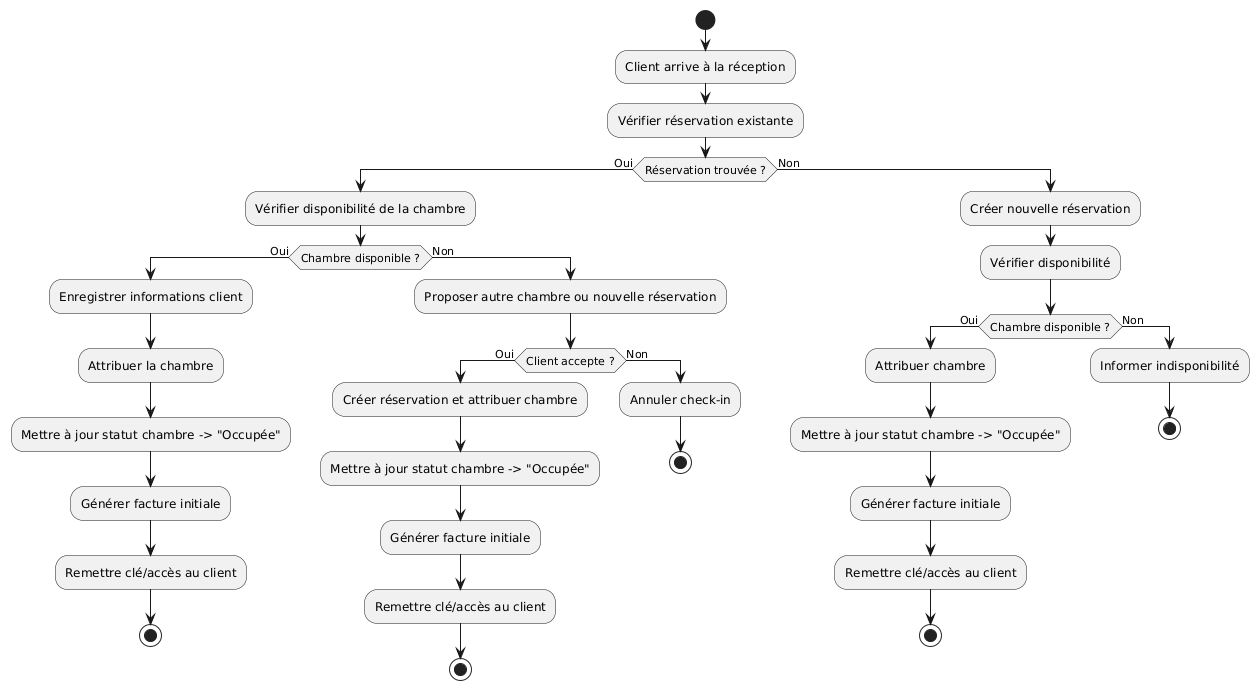
## 3. Diagramme de classe



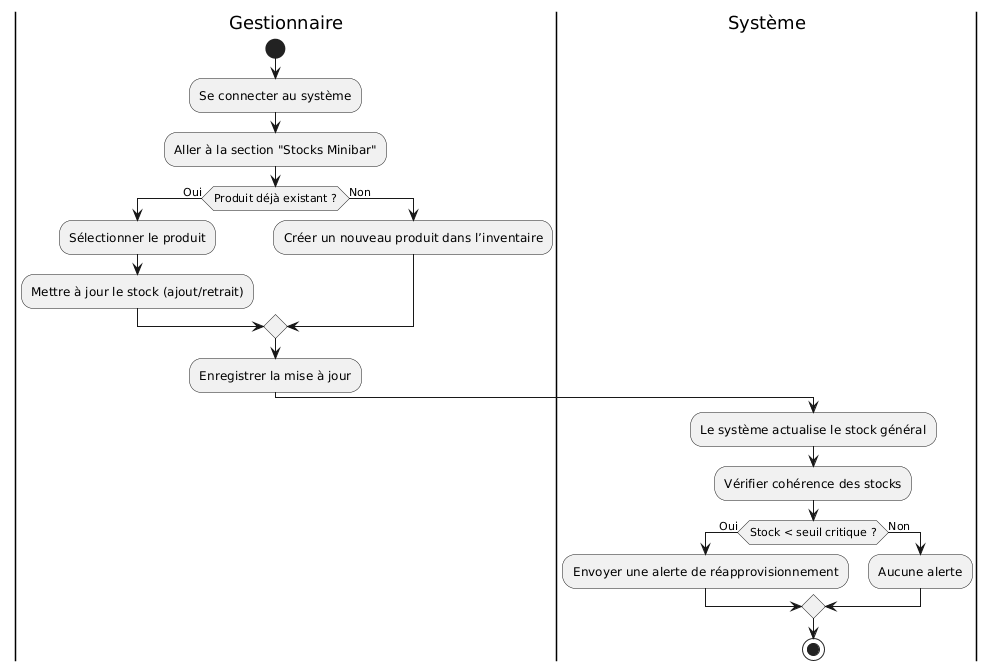
Lien PlantUML : [LIEN](https://www.plantuml.com/plantuml/uml/bLRDJjj04BxxAQnS6WH81JM78hKYIegA12WDALUqSIR9LkjTCxjRKLLzd-OvycBThEFfk7PZu1BnpSzyypEplhHEeFEvIjwa7mxz1VbukrbABPqqEbs0JJCbh0s_VIVcAlmUf_bsuzOco8uwtINHEfsQdUNF90r_ntBnYE7_z2Azlfj5ICavg8Wy3AUZpyDfV_3sA4fEJqkR305mvwLVuiUvMSYbh6FM5mNAdC1VLVXx9vntZzbQv7C4niD6mUkOh8LU0POkgyXYkPJPcg4K-afapcCj-MK9iQZQ1zBcGD0LQ430qXdbcmcLUGKuCqueelBDU0Tu8rUQ5E6vC1Z0oVYASuiw5Hr6K0MSqmjGWMRXfQk1LCycxmBbSu37dqRNjyER_jdWvBmKp1w6BJWbIYdT6s_hQHM5aayUC7ZWESn-5D3IMCVXrxHt0RlXn-2sCMfxZ4AFX8ELGSdZPeIzF2ZrZ1pM8pS5DEbCmecLULlEAq87P-wietV4DdW6_0vOaTdbXu3_xi8Od93A1fZlpIHuvro_Egca5GuyShx-BymvcxnrsA1T07aWq88xK52iZTRGahvO85ZB9UU3GSin6fcyS8rEJT0ifMgqgQviORGBdZb0NefRgULS8Aj27X6gG7DY9kLQLeuHdCGQoXiIWZhGdeVSQ_BAqwPtpszZLDwVlEjUytRwl8WSvbNaEDsASgHcir5RNbJ1ChydBxIJ3lvVq2zxZNf52UOngZK8lfilLPG99uscBcZjLPk4-iD5B4vHvL8_r4yEEkU2tWkUJ5COGdL_5an1HMesxMTgkx7fTkCmhFJsDxxeGw2swBsxNg3yCUsTzVRV1XVf_hCWeUl5bMRrUrw_yEqs_DKdAnknf5gre4Rgv6EdI9vfGxZd4Rlo_EWdzHULcyHga_UzHjAOKswo9najAeN35aJsWhtP2GqgLstFjbMLTowgy-lCJ3gpRKnoD_UcIMUKIz0Bdwi_)

## 

## 4. Diagramme d’activité (check-in client)



Lien PlantUML : [LIEN](https://www.plantuml.com/plantuml/uml/rPF1ZjGm38RlUGfhJxE75NSgmQ8HGXnWqHwuPp8kQqYJod56F0v7-XnvCToCjXjKYbPpsnxQDBR_n9_jsoIM9G_Ur8Knh_UUC0XOPZeXb5_WBN2P78v2CNJgyRLCJ3qXp_i9-MHd2-1FKeqWs1dgONVVseHZFfK9uUqro0C6sDrbkZQWJoDtf3J6G0Vo96M28yvdkmSx71YxwZmBxyyRYxV_bsfLVXyOlr4ILcqAVUIXtYQ1gpbsJvxlH9WE6RaviJ5_GhLN6DzZPj0q9SkZ7zoyWQixv_Ae2LurKH_A53Gt5UsjawoU54Z8-bRw7eUpkFDbUcMTAxyJsBowOf8urX_q2M7tEOOcqIySnwYqDL2bbel5346ve_VuLwaMrShpiU1kBl4MothCNaLMLTV81TwAtENqdatmsHHN93TedXiYXEnhHkX-t53udqGuKc-ULeraIssx4jiZS47JDwtyylhuOntIEgqh44ry2xk-pQr-y-3D7m00)



Lien PlantUML : [LIEN](https://www.plantuml.com/plantuml/duml/PP8zZjH048NpFaLf8qz4RYEnAxGYOG794ddJhjcjeLqzL5LRZEI0Qv0HudFu9fo4RWyOYSswVjuhlq--LVDYUOnkVeDgb9Wz2SvkwplNzWYXD34O2lWCUb5RVuxOkVOknj9RVq3qe1Yg66vwI-6pmWCnVV9oqpb7HsW-I1eo6GphSYe2_4h5dmrU7S2UaA5vd-dW0Df-NUBLYb4W8fologuE7z1Ci5uyfRnDjLw3nfVQNWYQUB93vp0gGlCkyURvMjO51J83fpoXptzDOV2i47zz-quy8LjDNiGyqB4alsV1Hr8Jb9fm9DrFbrHp_mV4xDgtk6C17opxM7VtptjS5ruNyR4m-rWUT2IKGlMfl943me1wNTLkexKnX9UWc2b243Bwal4vh7kUqcLBXk0ZYbKhA6dzkIISI3UC8xBzZ-GkXt-YFR9QEhlRKjJ_uJS0)

# 

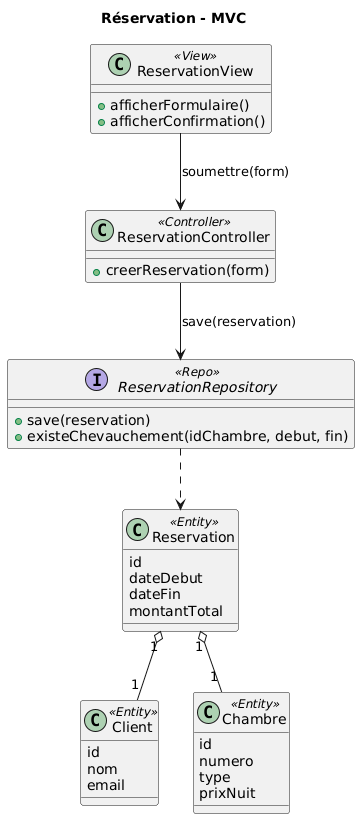
# 

# IV - Design Patterns

## Patterns utiles pour ce projet

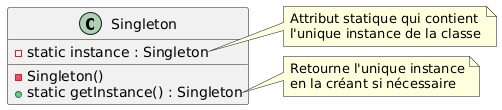
[MVC](https://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa70000)

On a choisi MVC pour séparer clairement l’interface utilisateur, la logique métier et l’accès aux données. Cette séparation facilite l’évolution du code et les tests unitaires sans impacter l’ensemble de l’application. Elle permet aussi d’avoir une structure standardisée et maintenable, adaptée à un projet multifonctionnel comme BlueMoon.



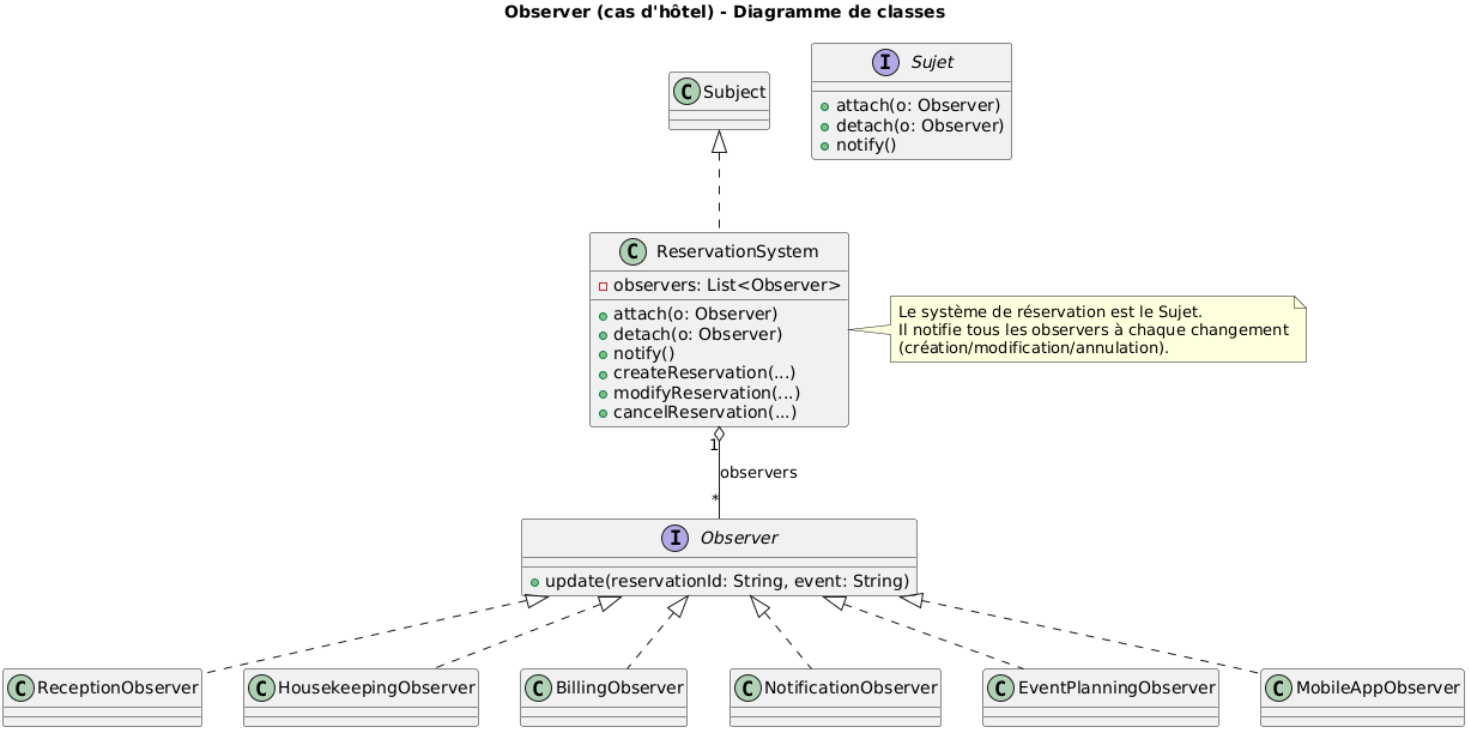
[Singleton](https://www.plantuml.com/plantuml/uml/VL0zqi8W4EqvnNPVCjzuWLInjDKJ85dZpk0oWQLoF53EaOj9Y0ocaMhPz_CUz46qb_YmobWT0bo8HulY69uAqZb08WWP84u36uIkSZP6lJTjtlmNpOXo-iYQTYTyASLE43oDTm5tgr3NbQJiTHJnT8soMKuHOOe4nh4GicIA_Oky8hNYW60rv0UXGXvWpVgHM7gMq7DQHy-uiwumShRtowmv5IFWPJOOWYR_5TUdQVtNDm00)

Nous avons aussi choisi d’implémenter la classe d’accès à la base de données en singleton afin de centraliser la gestion de la connexion, éviter les multiples instances coûteuses et garantir la cohérence des accès.



[Observer](https://www.plantuml.com/plantuml/uml/fLDBZjGm4Dtx5ALiI0ET4TlM20q894OQFg9Fu7PUTniSEzYLaKQ0n1su04lOSuE-2IV1yNGy3SvkLgcyz_ogNiMv22mz3vqHhDc0tcu2_3KyrKe6QX_i3xyPPa5_ldsd5rhklEmwK0jIHeQ088Is3B-L2hGUDX-Wc3uBeaUIMQfzxLRPSZ72BMPXwrXlR-g5-7fYc2MtZaFVIaRjCOAIjRENxOhMxBNTFIPSm_BqcdpIXFJ-Jhw-2OmkcItTqJgiw4e7FfzQFRt7xB5M7f9nqhDkcYOndMkZQevHqYgOWZb9eD2F-DJrYBzoGy17e8-1_wEUQsDAzCqug5PonklbkBrtHbfR7dljDjhWMTzdN4nV-lnBqvGR5fCmyUNy__9bZaAIyyopPQv2KkOhD7CvoyjJFQd8BPTKFQme7rzHlav2MCSWhtTx9hUTMSuLACJgyFFsF_A7Nrb124m6L1ttMpNYqf1DyK0SznJPSDUC3Zz8xUMd0UF3xj31igXLz4n-PvrhysREOlZ1f7BH2DXsD8QuYDNGcRy0)

L’Observer est pertinent pour un hôtel car il permet de synchroniser automatiquement tous les services (réception, ménage, facturation, notifications, etc.) dès qu’une réservation change, sans qu’ils dépendent directement les uns des autres.



# V - Documentation

## 1. README

Readme disponible sur [GitHub](https://github.com/livecampus-projects/blue-moon)

## 2. Rétroplanning de la conception

* ~~J1 matin : Analyse besoins + dictionnaire~~
* ~~J1 aprèm : Fonctions, MCD~~
* ~~J2 matin : MLD→MPD + choix SGBD, index~~
* ~~J2 aprèm : Début UML (Use cases + séquence + activité)~~
* ~~J3 matin : Mise en place des diagrammes~~
* ~~J3 aprèm : Refonte des premiers schémas UML~~
* ~~J4 matin : Design patterns~~
* J4 aprem : Schema UML des patterns / Consolidation & doc (README + slides)

## 3. Planning prévisionnel dev

| **#** | **Étape** | **Durée (j)** | **Dépendances** | **Livrables** |
| --- | --- | --- | --- | --- |
| 0 | Ouverture du projet, définition des besoins et planification globale | 1 | - | Vision partagée, liste des besoins classés par priorité, critères de qualité |
| 1 | Mise en place du socle technique (dépôt de code, CI/CD, outils qualité, base de données) | 2 | 0 | Pipelines automatiques, gestion des environnements (dev/test), base initiale |
| 2 | Gestion des utilisateurs et des rôles (connexion, sécurité, droits d’accès) | 3 | 1 | API d’authentification, écrans de connexion, règles de mot de passe |
| 3 | Gestion des bases principales : Clients & Chambres (ajout, modification, recherche) | 3 | 1 | API + interface utilisateur, validations, données de test |
| 4 | Disponibilités & moteur de réservation (règles, conflits) | 4 | 3 | Services de gestion des dispos + tests unitaires et d’intégration |
| 5 | Interface réservation (choix des dates, attribution d’une chambre) | 3 | 4 | Écrans de réservation + parcours de base |
| 6 | Paiements (carte/espèces) & suivi des statuts | 4 | 5 | Intégration prestataire de paiement, gestion des statuts (autorisé, capturé, etc.) |
| 7 | Facturation (numéros, PDF, TVA, taxe de séjour) | 3 | 6 | Modèle de factures, génération de PDF, stockage des factures |
| 8 | Emails & notifications (confirmation réservation, modèles d’emails) | 2 | 6 | Modèles d’emails, envoi automatique à la confirmation |
| 9 | Gestion stock minibar (global) | 3 | 1 | CRUD produits, seuils, mouvements de stock global |
| 10 | Gestion stock minibar (global) | 4 | 9 | Tableaux par chambre/produit, suivi mouvements & écarts |
| 11 | Facturation minibar au check-out (ajout automatique à la facture) | 2 | 7,10 | Lignes minibar ajoutées aux factures + tests |
| 12 | Rapports de base (taux d’occupation, chiffre d’affaires par période) | 2 | 7, 10 | Vues SQL + écrans de synthèse |
| 13 | Tests de bout en bout, performances de base, sécurité | 4 | 5, 12 | Scénarios E2E, améliorations sécurité OWASP, optimisation & monitoring |
| 14 | Recette utilisateur (UAT) & corrections | 3 | 13 | Procès-verbal de recette, liste de corrections |
| 15 | Mise en production V1 & transfert | 1 | 14 | Version publiée, notes de version, guide d’exploitation |

## 4. Support de présentation (slides)

***Est-ce que Corentin a signé ?***

📝 Énoncer

Perf Concetption - TP – application de gestion d’hôtel

Vous allez travailler sur la conception d’une application de gestion d’hôtel. L’objectif est de suivre toutes les étapes de l’ingénierie logicielle :

1. Analyse des besoins.

2. Modélisation des données (Merise).

3. Modélisation fonctionnelle et dynamique (UML).

4. Intégration de design patterns.

5. Réalisation d’un mini-prototype.

Contexte

L’hôtel BlueMoon souhaite informatiser son système de gestion. Actuellement, tout est fait sur papier ou Excel, ce qui entraîne des erreurs et des pertes de temps.

L’application doit permettre de gérer :

Les clients et leur historique de réservation (une réduction est appliquée toutes les 10 réservations).

Les chambres et leurs caractéristiques (dont les minibars : leurs contenus, catégories de contenu, stock du minibar, stock général de l'hôtel).

Les réservations (création, modification, annulation).

Les paiements et factures.

Les utilisateurs (administrateur, réceptionniste, client).

Les premiers échanges ont permis de définir les userstories suivantes :

Gestion des clients

En tant que réceptionniste, je veux enregistrer un nouveau client avec ses coordonnées afin de pouvoir lui créer une réservation.

En tant que réceptionniste, je veux rechercher rapidement un client par son nom ou son email afin de retrouver son historique de séjours.

Gestion des chambres

En tant qu’administrateur, je veux ajouter une nouvelle chambre avec son type (simple, double, suite), son prix et son statut afin qu’elle soit disponible à la réservation.

En tant que réceptionniste, je veux voir les chambres disponibles pour une période donnée afin de proposer une réservation au client.

Gestion des réservations

En tant que client, je veux réserver une chambre en ligne pour des dates précises afin de garantir ma place.

En tant que réceptionniste, je veux modifier ou annuler une réservation existante afin de gérer les imprévus du client.

Gestion des paiements

En tant que réceptionniste, je veux enregistrer le paiement d’une réservation (espèces, carte, en ligne) afin de valider le séjour du client.

En tant qu’administrateur, je veux générer une facture associée à une réservation afin de la remettre au client.

En tant que gestionnaire, je veux avoir la liste des factures payées pour les stocks dédiés aux minibars.

Gestion des utilisateurs et notifications

En tant qu’administrateur, je veux créer des comptes utilisateurs avec des rôles différents (client, réceptionniste, admin) afin de sécuriser l’accès au système.

En tant que client, je veux recevoir un email de confirmation après avoir réservé une chambre afin d’avoir une preuve de ma réservation.

Gestion des stocks

En tant que gestionnaire, je veux voir les stocks généraux de l'hôtel par produits vendu dans les minibars.

En tant que gestionnaire, je veux mettre à jour les stocks suite à une commande ou une recharge de minibars

Travail à faire

✅ 1 – Analyse des besoins

Lister les acteurs.

Décrire les fonctions principales du système :

réserver une chambre,

gérer les clients,

gérer les chambres,

enregistrer un paiement,

générer une facture.

Produire un dictionnaire des données du cas à traiter

✅ 2 – Modélisation des données (Merise)

Construire un MCD avec les entités principales : Client, Chambre, Réservation, Paiement, Facture, etc.

Définir les associations et les cardinalités.

Transformer le MCD en MLD

Selectionner le moteur de base de données le plus à même de répondre aux besoins de l'app Transformer le MLD en MPD

3 – Modélisation UML

✅ Réaliser les cas d’utilisation (use case).

✅ Créer un diagramme de séquence (exemple : réserver une chambre).

✅ Réaliser un diagramme de classes du système.

✅ Créer un diagramme d’activité (exemple : check-in client).

4 – Design Patterns

Identifier 3 à 4 patterns utiles pour ce projet.

Exemple :

MVC : séparer interface / logique / données.

DAO : gérer l’accès aux données.

Strategy : calcul du prix selon la saison.

Observer : notifier le client quand sa réservation change.

Illustrer chaque pattern choisi par un diagramme UML (Diagramme de classe ou de package).

5 – Documentation

Vous réaliserez une documentation qui comprendra :

Un Readme qui fera état des généralités de l'app et toute information pertinente à destination du développement

✅ Un rétroplanning de la conception

✅ Un planning prévisionnel du développement

✅ Un support de présentation de quelques slides qui illustre votre phase de conception (besoins, plannings, Merise, UML)

Déroulé du projet

Dans un premier temps, vous ferez une analyse du sujet

Vous aurez alors l'occasion de poser toutes les questions qui vous permettraient d'avancer dans l'analyse des besoins et l'établissement du dictionnaire des données.

Nous ferons une mise en commun du MCD Mercredi à 11h00 (plus tôt si toutes les équipes ont terminé). Vous pourrez alors décider de continuer avec votre solution ou la solution commune.

Idéalement, le passage au MLD puis MPD sera terminé mercredi soir.

Par la suite vous ferez une analyse UML du projet qui devrait être terminée jeudi soir. Vous traiterez ensuite les design patterns pour isoler quelque patterns intéressant avec une représentation en diagramme de classe. Ils seront idéalement traités vendredi midi. Enfin vous réaliserez la documentation du projet.

Rendu

Le tout sera à rendre pour le dimanche 21 Septembre à 23h59 dernier délai (5 pts en moins par demie journée de retard). Votre projet sera zippé et sera nommé avec vos noms et prenoms. Ce zip sera partagé via NetYparéo. Idéalement, votre zip ne contiendra pas de fichiers liés à des logiciels propriétaires (quels que soient les logiciels utilisés, précisez leur nom dans un README dans votre zip).