1. **Introduction :**

La conception d’un projet est une phase très importante pour définir les objectifs et les fonctionnalités de notre application. Dans ce chapitre nous nous intéressons à l’étude de conception de notre application, nous adoptons l’UML comme langage de modélisation.

Cela consiste à présenter le diagramme de cas d’utilisation décrivant les scénarios nominaux de chaque acteur ainsi que le diagramme composant qui représente visuellement les relations entre les différents composants de notre système et le diagramme de déploiement qui  est utilisé pour visualiser les processeurs matériels, les nœuds et les dispositifs d’un système.

1. **L’objectif :**

Pour bien comprendre la technologie de Next.Js, nous avons opté la réalisation d’une application web qui répondre aux attentes suivantes :

* + - Développer un plan d’étude avec une bonne analyse du problème posé.
    - Planifier les différentes étapes de conception du projet.
    - Réaliser une application englobant le maximum de tâches fonctionnelles avec des solutions optimales.
    - Permettre à l’administrateur de gérer la base de données.
    - Assure la sécurité de l'application.

1. **Spécification des besoins :**

La phase de spécification des besoins est la première phase formelle obligatoire dans le développement d’une application informatique puisque la capacité de persuasion d’un produit ne peut se réaliser parfaitement sans une spécification préalable élaborée des besoins et des exigences.

* 1. **Les besoins fonctionnels :**

Les besoins fonctionnels servent à présenter les actions que doit effectuer le système en réponse à une demande présentée par un utilisateur.

Notre application doit répondre aux exigences suivantes :

* L’application doit permettre aux clients de consulter le site ainsi que rechercher des chambres sans l’authentification.
* Pour réserver une chambre le client doit s’authentifier.
* Le client peut annuler ça réservation.
* Administrateur gère les comptes des clients.
* L’application doit permettre à l'administrateur de gérer les chambres avec la possibilité d’ajouter ou supprimer ou modifier une chambre.
  1. **Les besoins non-fonctionnels :**

Les besoins non fonctionnels décrivent les objectifs liés aux performances du système et aux contraintes de son environnement. Ses exigences techniques sont souvent exprimées sous forme d’objectifs spécifiques que doit atteindre le système.

* **La sécurité :** L’application doit tenir compte de confidentialité des données des utilisateurs.
* **La performance :** Temps de réponse court (une réponse aux requêtes doit être rapide pour minimiser le temps d’attente de l’utilisateur).
* **La fiabilité :** Minimiser des risques et des erreurs afin d’assurer le bon fonctionnement du système.
* **Ergonomie et convivialité :** L’application doit fournir une interface simple et élégante pour l’utilisateur afin de faciliter l’exploitation des services de l’application.
* **Guidage :** L’ensemble des moyens mis en œuvre pour orienter, informer les utilisateurs lors de ses interactions avec l’application.

1. **Langage de modélisation :**

On a utilisé dans ce projet comme langage de modélisation l’UML.

* 1. **UML :**

**U**nified **M**odeling **L**anguageou langage de modélisation unifié est un langage de modélisation de développement à usage général dans le domaine du génie logiciel qui vise à fournir un moyen standard de visualiser la conception d’un système

* 1. **Les diagrammes d’UML :**

UML propose 13 diagrammes regroupés dans deux grands ensembles :

1. **Les diagrammes structurels :**
   * Diagramme de classe.
   * Diagramme d’objet.
   * Diagramme de composants.
   * Diagramme de déploiement.
   * Diagramme de paquetages.
   * Diagramme de structures.
2. **Les diagrammes de comportement :** 
   * Diagramme des cas d’utilisation.
   * Diagramme d’état-transition.
   * Diagramme d’activités.
   * Diagramme de séquences.
   * Diagramme de communication.
   * Diagramme global d’interaction.
   * Diagramme de temps.
3. **Les diagrammes utilisés :**

Pour mieux concevoir notre application, nous avons besoin d’utiliser les trois types de diagrammes suivants :

|  |  |  |
| --- | --- | --- |
| **Diagramme** | **Objectifs** | **Type** |
| Diagramme De  Cas d’utilisation | 1. Identifier la fonctionnalité du système. 2. Il décrit l'interaction des personnes ou du dispositif externe avec le système en cours de conception. 3. Il résume les relations entre les cas d'utilisation, les acteurs et les systèmes. | Fonctionnel |
| Diagramme de composants | 1. Il permet aux concepteurs d'applications de vérifier que les fonctionnalités requises d'un système sont mises en œuvre par les composants, garantissant ainsi que le système final sera acceptable. 2. De plus, le diagramme des composants est un outil de communication utile entre les parties prenantes pour discuter, analyser ou améliorer la conception du système. | Statique |
| Diagramme de déploiement | 1. Décrire les composants matériels utilisés dans les implémentations de systèmes ainsi que les environnements d'exécution et les artefacts déployés sur le matériel. 2. Il permet de visualiser le système de topologie du matériel, de modéliser les éléments matériels physiques et la relation de communication entre eux, et de planifier l'architecture du système. | Statique |

1. **Diagrammes des cas d’utilisations :**

Un diagramme de cas d’utilisation dans sa forme la plus simple est une représentation de l’interaction d’un utilisateur avec le système qui montre la relation entre l’utilisateur et les différents cas d’utilisation dans lesquels l’utilisateur est impliqué. Un diagramme de cas d’utilisation peut identifier les différents types d’utilisateurs d’un système et les différents cas d’utilisation et seront souvent accompagnés également d’autres types de diagrammes.

Les diagrammes de cas d’utilisation contiennent généralement :

• Cas d’utilisation.

• Acteurs.

• Relations de dépendance, généralisation et association.

Comme tous les autres diagrammes, les diagrammes de cas d’utilisation peuvent contenir des notes et des contraintes. Ils peuvent également contenir des packages, qui sont utilisés pour regrouper des éléments de votre modèle en blocs plus volumineux. Parfois, vous souhaiterez également placer des instances de cas d’utilisation dans vos diagrammes, en particulier lorsque vous souhaitez visualiser un système d’exécution spécifique.

* 1. **Identifications des acteurs :**

Avant d’entamer la présentation des diagrammes, il faut identifier les acteurs qui interagissent directement avec le système. Un acteur est une entité externe qui peut être un utilisateur humain, un dispositif matériel ou un autre système.

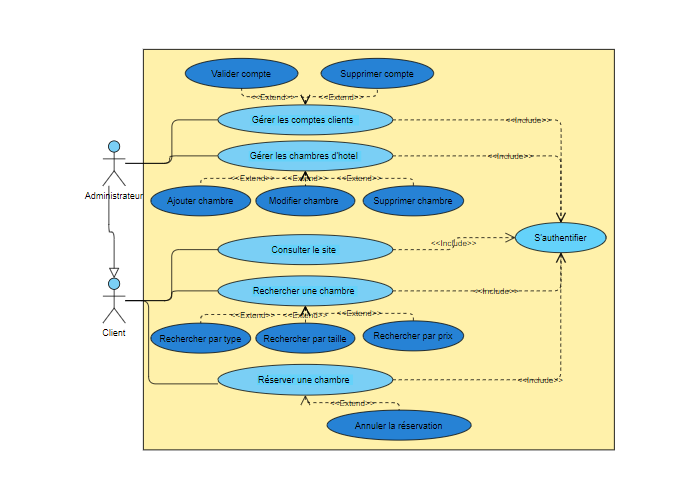
Dans notre système on distingue deux acteurs principaux

* Administrateur : c’est lui qui va gérer les comptes clients ainsi que les chambres d’hôtel (Ajouter, Modifier et Supprimer).
* Client : Peut consulter le site ainsi que rechercher des chambres, et faire des réservations de chambres selon son choix.
  1. **Identification des cas d’utilisation :**

Un cas d'utilisation (en anglais use case) permet de mettre en évidence les relations fonctionnelles entre les acteurs et le système étudié. Le diagramme de cas d'utilisation permet de représenter visuellement une séquence d'actions réalisées par un système.

Voici quelques cas d’utilisation de notre système :

* Gérer les comptes clients. Gérer les chambres.
* Rechercher une chambre, Réserver une chambre….
  1. **Diagramme de cas d’utilisation globale de notre application :**

****

1. **Diagrammes des composants :**

Décrivent les composants du logiciel, leurs interfaces et leurs dépendances.

Les diagrammes de composants sont utiles pour les raisons suivantes :

* Définition des aspects exécutables et réutilisables d'un système logiciel.
* Mise en évidence des problèmes de configuration logicielle à travers les relations de dépendance.
* Représentation précise d'une application logicielle avant d'y apporter des

Changements ou des extensions.

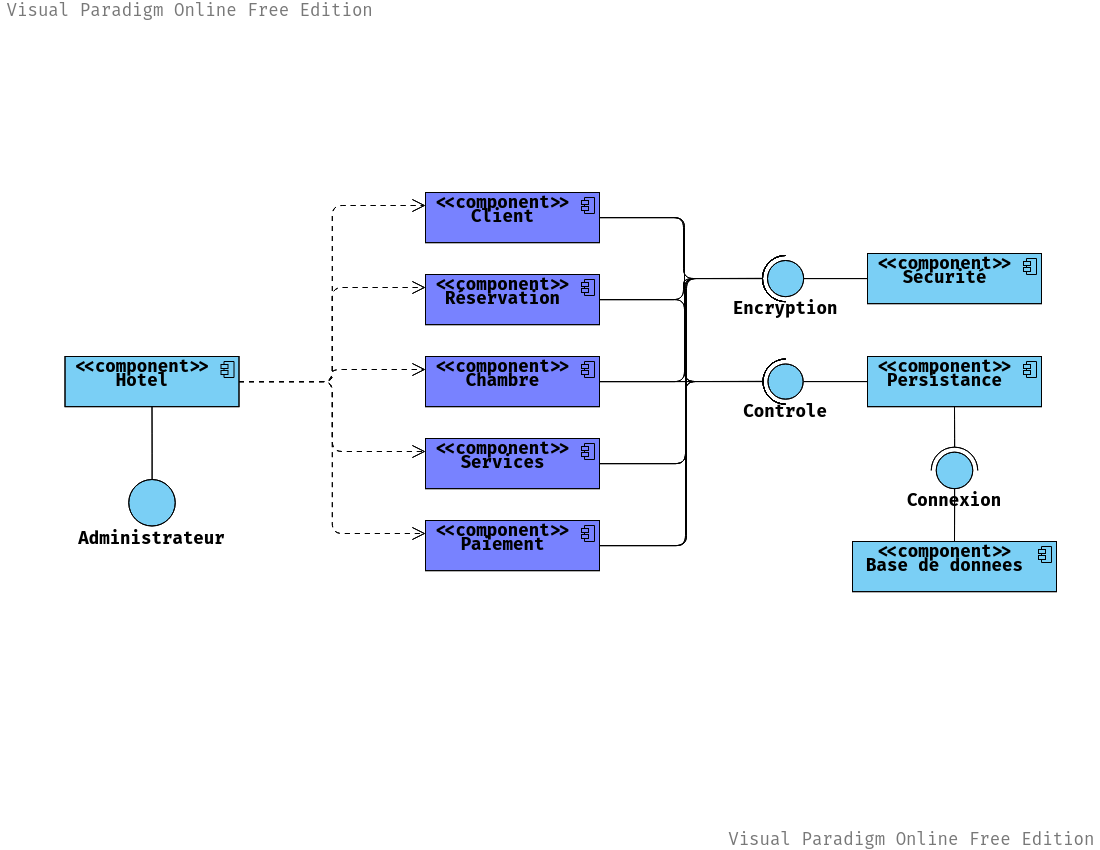
* 1. **Un composant :**

Chaque composant est assimilé à un élément exécutable du système. Il est caractérisé par :

* un nom.
* une spécification externe sous forme soit d’une ou plusieurs interfaces requises1, soit d’une ou plusieurs interfaces fournies.
* un port de connexion.

Le port d’un composant représente le point de connexion entre le composant et une interface. L’identification d’un port permet d’assurer une certaine indépendance entre le composant et son environnement extérieur

* 1. **Diagramme de composant global :**

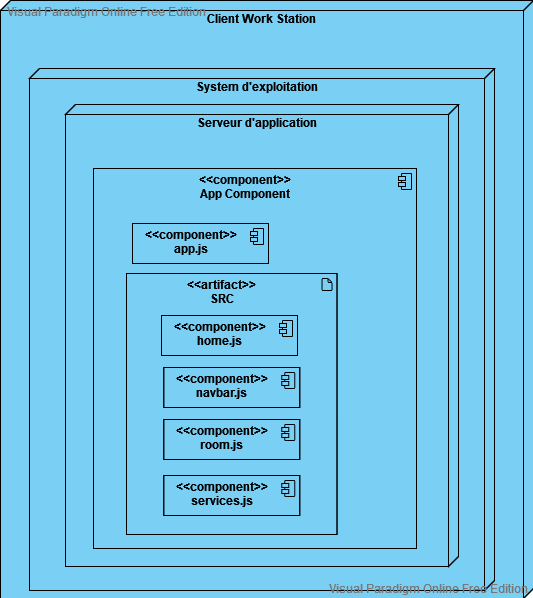
****

1. **Diagrammes de déploiement :**

Un diagramme de déploiement dans le langage de modélisation unifié modélise le déploiement physique des artefacts sur les nœuds. Pour décrire un site Web, par exemple, un diagramme de déploiement montrerait quels composants matériels «nœuds» existent (par exemple, un serveur Web, un serveur d’applications et un serveur de base de données), sur quels composants logiciels «artefacts» s’exécutent chaque nœud (par exemple, application Web, base de données), et comment les différentes pièces sont connectées (par exemple JDBC, REST, RMI).

Les nœuds apparaissent sous forme de cases et les artefacts alloués à chaque nœud apparaissent sous forme de rectangles dans les cases. Les nœuds peuvent avoir des sous-nœuds, qui apparaissent comme des boîtes imbriquées. Un seul nœud dans un diagramme de déploiement peut représenter conceptuellement plusieurs nœuds physiques, tels qu’un cluster de serveurs de base de données.

* 1. **Diagramme de déploiements de notre application :**

****

1. **Conclusion :**

Dans ce chapitre, nous avons défini d’abord la signification de l’analyse et la conception avec les outils qu’on a utilisés pour réussir notre conception, ensuite nous sommes passés aux différents diagrammes là où nous avons expliqué notre solution en détail.

Cette étude nous a permis de définir la structure de l’application et cela aussi nous a facilité l’implémentation que vous verraient dans le chapitre suivant.