Introduction à la modélisation conceptuelle de données avec UML

Table des matières

I. Contexte **3**

II. Modélisation conceptuelle de données **3**

A. Conception d'une application 3

B. UML 5

III. Travailler avec l'UML **7**

A. Débuter le travail avec UML 7

B. Exemples de diagrammes 8

IV. Essentiel **11**

V. Auto-évaluation **12**

A. Exercice 12

B. Test 13

Solutions des exercices **13**

I. Contexte

Durée : 1 h

Environnement de travail : Draw.io / Visual-Paradigm

Prérequis : aucun

Contexte

Lorsque l'on commence à apprendre le développement, on pourrait croire que la capacité de coder est la seule capacité à intégrer. Pourtant, le codage n'est qu'une partie de la création d'un logiciel. En effet, l'informatique existe pour rendre service à des humains.

Il faut donc en premier lieu comprendre le besoin d'un utilisateur et le transformer en outil informatique, et ensuite seulement, nous pourrons créer l'outil informatique adapté.

Un site codé proprement n'est pas forcément le meilleur outil pour l'utilisateur, il peut ne pas être pratique pour l'utilisateur, ou du moins, ne peut répondre que partiellement à son besoin. Il est donc fondamental, avant de commencer le développement, de chercher à produire le site le mieux adapté à la demande client.

Dans ce cours, nous verrons les étapes qui précèdent le codage dans la construction d'un logiciel, et nous parlerons d'une des clés de ces étapes : l'UML.

II. Modélisation conceptuelle de données

A. Conception d'une application

Exemple **La recette**

Imaginons que vous soyez un chef restaurateur, et que vous souhaitez expliquer votre recette à d'autres cuisiniers. Pour cela, vous allez devoir détailler votre recette et l'écrire de façon claire et précise, afin que les cuisiniers puissent l'appliquer. La phase de conception consiste en la même chose. L'objectif est de comprendre le produit informatique, de détailler ses composants, et de spécifier comment les imbriquer les uns avec les autres. Un plat est bien réussi si sa recette est bien faite. De même, un logiciel est bien réussi si sa modélisation est bien faite.

Fondamental **Conception d'une application**

Le processus de création d'un logiciel comporte plusieurs étapes. Le but est de partir d'un besoin du client, de transformer ce besoin en produit informatique, de détailler ce produit informatique, et ensuite de le réaliser. Avant le développement d'une application, on a donc plusieurs étapes consistant à définir le logiciel à développer. Le but sera de définir le périmètre fonctionnel de l'application, et de définir les contraintes techniques.

On peut donc distinguer quatre étapes lors de la conception d'une application :

- Récupération du besoin client
- Spécifications fonctionnelles (définition des modules à développer, et des règles à respecter)
- Spécifications techniques (définition des contraintes du développement)
- Développement (codage du projet)

Spécification fonctionnelle

Une spécification fonctionnelle est un document détaillant les fonctionnalités et les comportements attendus d'un produit, d'un système logiciel ou d'une application. Elle décrit les différentes actions, tâches ou opérations que le système doit être capable d'effectuer pour répondre aux besoins et aux attentes des utilisateurs.

L'objectif principal d'une spécification fonctionnelle est de fournir une description claire et complète des fonctionnalités souhaitées, afin de guider le processus de développement et de s'assurer que le produit final répond aux besoins des utilisateurs. Elle permet également de faciliter la communication et la collaboration entre les différents acteurs du projet, en fournissant un cadre de référence commun pour les discussions et les prises de décision.

Spécification technique

Les spécifications techniques, également appelées spécifications techniques des besoins, sont des documents qui décrivent de manière détaillée les exigences et les caractéristiques techniques d'un produit ou d'un système. Elles fournissent des informations spécifiques sur les composants, les fonctionnalités, les performances, les normes et les contraintes techniques nécessaires à la conception et au développement du produit ou du système.

Ce type de spécification peut inclure des détails tels que les spécifications matérielles, les langages de programmation, les interfaces, les protocoles de communication, les exigences de sécurité, les performances attendues, les tests à effectuer, etc. Elles aident à aligner les attentes entre les différentes parties prenantes et à garantir la conformité et la qualité du produit final.

Exemple **Demande de développement d'un logiciel**

Imaginez que le chef d'un restaurant contacte une équipe de développement pour réaliser le site vitrine de son restaurant. Une bonne conception du site pourrait suivre ces étapes :

1. **Récupération du besoin client :** l'équipe de développement rencontre le propriétaire du restaurant pour discuter de ses besoins et attentes concernant le site web. Le propriétaire explique qu'il souhaite un site web attrayant et convivial qui présente le menu du restaurant, permettant aux clients de réserver une table en ligne et affichant des informations sur les horaires d'ouverture, l'emplacement et les spécialités du restaurant. L'équipe pose des questions supplémentaires pour obtenir des détails spécifiques, comme le style de design préféré, les fonctionnalités particulières requises et les contraintes budgétaires.
2. **Spécifications fonctionnelles :** sur la base des discussions avec le client, l'équipe élabore les spécifications fonctionnelles du site web du restaurant. Ces spécifications comprennent la conception d'une page d'accueil attrayante avec une galerie de photos mettant en valeur les plats du restaurant, une section dédiée au menu où les utilisateurs peuvent consulter les différentes catégories de plats et leurs descriptions, un système de réservation en ligne avec un formulaire de réservation et un calendrier pour vérifier la disponibilité des tables, une page d'informations fournissant des détails sur l'emplacement, les heures d'ouverture et les coordonnées du restaurant, ainsi qu'une section pour les témoignages clients.
3. **Spécifications techniques :** les spécifications techniques définissent les contraintes et les exigences techniques pour le développement du site web du restaurant. Par exemple, elles indiquent l'utilisation du langage HTML, CSS et JavaScript pour la partie front-end, et PHP avec une base de données MySQL pour la partie back-end. Elles précisent également que le site web doit être compatible avec les principaux navigateurs tels que Chrome, Firefox et Safari, et qu'il doit être adaptatif, c'est-à-dire qu'il doit s'ajuster automatiquement à différentes résolutions d'écran, y compris sur les appareils mobiles. Les spécifications techniques peuvent également inclure des exigences de sécurité, comme l'utilisation de certificats SSL pour les transactions en ligne sécurisées.

Ainsi, l'équipe de développement a récupéré les besoins du client, a défini les fonctionnalités du site web à travers les spécifications fonctionnelles, et établi les contraintes techniques pour le développement à travers les spécifications techniques. Il ne reste plus qu'à développer le site maintenant.

Est-ce vraiment utile toutes ces étapes ?

En plus de dire que c'est utile, nous pouvons même dire que c'est fondamental. Ces étapes nous permettent d'éviter les mauvaises surprises au cours du développement. Le but est que le client ait une vision claire et précise du rendu final, avant même de commencer le développement. Nous évitons ainsi les mauvaises surprises. Une fois que le développement est lancé, annuler ou modifier des éléments est très coûteux.

Complément **Gestion de projet**

Ce cours parle de conception d'application, une étape fondamentale de la gestion de projet. Mais il existe différentes méthodes de gestion de projet (agile, cycle en V, etc.). Nous n'en parlerons pas dans ce cours. La modélisation fonctionnelle et technique est un point commun des méthodes de gestion de projet. Donc, quelle que soit l'organisation de l'entreprise dans laquelle vous travaillerez, ces modèles de conception vous seront très utiles.

B. UML

Rappel

Nous venons de voir qu'avant le développement, il y a différentes étapes de conception. Ces étapes de conception sont matérialisées par des documents textuels et/ou des schémas. Pour que ces schémas soient clairs nous devons tous avoir la même manière de les faire, et avoir une syntaxe commune. C'est là qu'intervient l'UML.

Définition **UML**

UML est un langage graphique utilisé pour représenter visuellement les systèmes logiciels. Il propose des symboles et des schémas standardisés qui aident les développeurs à comprendre et à communiquer sur la structure, le comportement et les interactions d'un logiciel. C'est un outil précieux pour la conception et la documentation des projets informatiques.

Histoire de l'UML

L'histoire de l'UML remonte aux années 1990, lorsque trois experts de renom en génie logiciel, Grady Booch, James Rumbaugh et Ivar Jacobson, ont travaillé de manière indépendante sur leurs propres méthodes de modélisation. Grady Booch a développé le Booch Method, James Rumbaugh a créé le OMT (Object Modeling Technique), et Ivar Jacobson a introduit le OOSE (Object-Oriented Software Engineering). Ces méthodes avaient toutes des points forts, mais elles manquaient d'uniformité et de compatibilité.

En 1994, les trois experts ont décidé de combiner leurs approches pour créer un langage de modélisation unifié qui tirerait parti des meilleures pratiques de chacune des méthodes. Ainsi est né l'UML (Unified Modeling Language), qui a été présenté pour la première fois en 1995 lors de la conférence Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA).

La création de l'UML a été soutenue par de nombreux acteurs de l'industrie du logiciel, et en 1997, l'Object Management Group (OMG), une organisation internationale dédiée à la standardisation des technologies de l'information, a adopté l'UML comme un standard. Depuis lors, l'UML a continué à évoluer et à s'enrichir avec de nouvelles versions et extensions pour répondre aux besoins croissants du génie logiciel.

Aujourd'hui, l'UML est devenu un langage de modélisation largement accepté et utilisé dans l'industrie du développement logiciel. Il fournit une gamme de diagrammes et de notations qui permettent de représenter de manière graphique la structure, le comportement et les interactions d'un système logiciel. L'UML est utilisé pour la conception, la communication, la documentation et l'analyse des systèmes logiciels complexes, facilitant ainsi le travail collaboratif entre les différents acteurs d'un projet et favorisant une meilleure compréhension des spécifications et des exigences du système à développer.

Types de diagrammes

L'UML définit 14 diagrammes séparés en deux types : les diagrammes de structure et les diagrammes de comportement.

Les diagrammes de comportement permettent de définir les fonctionnalités et les interactions dans une application. Ces diagrammes permettent de répondre à la question : « *Que va-t-on faire ?* ». Ils sont donc surtout utilisés dans les spécifications fonctionnelles.

Les diagrammes de structure permettent de définir les différents composants de l'application : les classes, les packages, les objets, etc. Ces diagrammes permettent de répondre à la question : « *Comment va-t-on faire ?* ». Ils sont donc surtout utilisés dans les spécifications techniques.

Exemple **Diagrammes de structure**

Les diagrammes de structure UML mettent l'accent sur la représentation des éléments statiques d'un système logiciel, tels que les classes, les objets, les composants et les relations entre eux. Ils permettent de visualiser la structure interne du système et les interactions entre les différentes entités. Voici quelques exemples de diagrammes de structure en UML :

- Diagramme de classes : il représente les classes du système, leurs attributs, leurs méthodes et les relations entre les classes (comme l'héritage ou l'association).
- Diagramme d'objets : il montre les instances spécifiques d'objets et leurs relations à un moment donné dans le système.
- Diagramme de composants : il illustre les composants du système, leurs interfaces, leurs dépendances et leurs relations.
- Diagramme de déploiement : il représente l'architecture matérielle du système et la configuration des nœuds matériels (par exemple, les serveurs, les ordinateurs) sur lesquels les composants logiciels sont déployés.

Exemple **Diagrammes de comportement**

Les diagrammes de comportement UML se concentrent sur la dynamique et le fonctionnement du système logiciel, en décrivant les interactions entre les différents éléments. Ils montrent comment le système réagit aux stimuli, comment les objets interagissent entre eux et comment les états du système évoluent. Voici quelques exemples de diagrammes de comportement en UML :

- Diagramme de cas d'utilisation : il représente les interactions entre les acteurs (utilisateurs, systèmes externes) et le système, en montrant les fonctionnalités offertes par le système du point de vue de l'utilisateur.
- Diagramme de séquence : il décrit la séquence chronologique des messages échangés entre les objets du système, mettant en évidence l'ordre d'exécution des actions.
- Diagramme d'activité : il modélise le flux d'activités et les actions dans un processus, en montrant les décisions, les branchements et les boucles.
- Diagramme d'états-transitions : il représente les différents états d'un objet ou d'un système, ainsi que les transitions entre ces états en réponse à des événements.

Ces diagrammes en UML offrent des perspectives complémentaires pour modéliser les différents aspects d'un système logiciel, en fournissant des outils visuels puissants pour la conception, la communication et la documentation des systèmes complexes.

Conseil

Après lecture de cette partie, vous pouvez vous sentir submergé par la quantité d'informations. Pas de panique ! C'est normal. Il ne vous est pas demandé de maîtriser tous les types de diagramme, mais d'en comprendre l'importance, et de savoir en réaliser quelques-uns. Ce n'est qu'avec la pratique et le temps que vous pourrez différencier et maîtriser ces diagrammes.

Généralement, il est demandé de maîtriser trois types de diagrammes fondamentaux : cas d'utilisation, séquence, et de classes.

III. Travailler avec l'UML

A. Débuter le travail avec UML

Les logiciels

Pour réaliser des diagrammes UML, vous aurez besoin d'un logiciel de diagramme. Vous pouvez aussi faire le diagramme sur papier, mais la maintenance de ce diagramme serait trop compliquée. Par exemple :

- Visual Paradigm : il propose à la fois une version payante et une version gratuite avec des fonctionnalités limitées. Il est convivial et offre une large gamme de fonctionnalités de modélisation UML.
- Lucidchart : c'est une plateforme de diagramme en ligne qui propose une version gratuite avec des fonctionnalités de base. Il existe également une version payante avec des fonctionnalités avancées et une collaboration en temps réel.
- draw.io : c'est un outil de diagramme en ligne entièrement gratuit, qui inclut la possibilité de créer des diagrammes UML. Il est convivial et offre une gamme de symboles UML.

Définition **UML et POO**

La Programmation Orientée Objet (POO) joue un rôle essentiel dans la modélisation avec UML. UML offre un ensemble de diagrammes qui permettent de représenter les concepts et les relations de la POO de manière visuelle et structurée. L'un des diagrammes clés pour la conception POO en UML est le diagramme de classes. Ce diagramme permet de représenter les classes, les attributs, les méthodes et les relations entre les classes. Il aide à visualiser la structure statique d'un système logiciel et à identifier les interactions entre les objets.

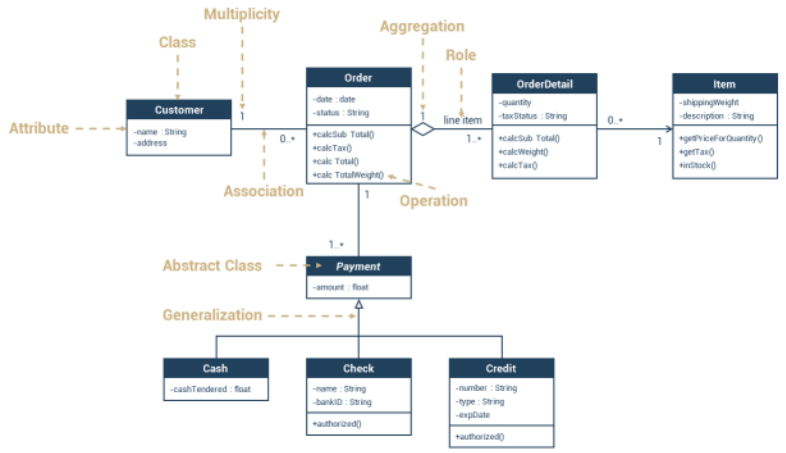


diagram-class-exemple

En utilisant le diagramme de classes, on peut définir les différentes classes du système, leurs propriétés et leurs méthodes. Les relations entre les classes, telles que l'héritage, l'association, l'agrégation et la composition, peuvent également être spécifiées. Cela permet de capturer les relations entre les différents objets et de modéliser la structure de l'application.

Outre le diagramme de classes, d'autres diagrammes UML liés à la POO existent :

- Les diagrammes d'objets
- Les diagrammes de séquence
- Les diagrammes d'états-transitions

La conception POO en UML favorise l'encapsulation, l'abstraction, l'héritage et le polymorphisme, qui sont des concepts clés de la POO. Ces principes permettent de créer des systèmes flexibles, modulaires et extensibles, en favorisant la réutilisabilité du code et la gestion efficace des fonctionnalités.

En utilisant UML pour la conception POO, nous avons une vision du développement de l'application, avant même de commencer. Cela nous permet de poser une structure fiable et bien faite, en prenant vraiment le temps de réfléchir. En effet, pendant le développement, nous avons souvent moins de recul. La création du diagramme nous permettra de penser notre structure de la manière la plus efficiente, en mettant de côté la « *paresse du développeur* ».

B. Exemples de diagrammes

Besoin utilisateurs

Reprenons l'exemple du restaurateur qui veut un site web de son restaurant dans la partie précédente. Voici les besoins utilisateurs :

- Présenter le menu du restaurant,
- Permettre aux clients de réserver une table en ligne,
- Afficher des informations sur les horaires d'ouverture, l'emplacement et les spécialités du restaurant.

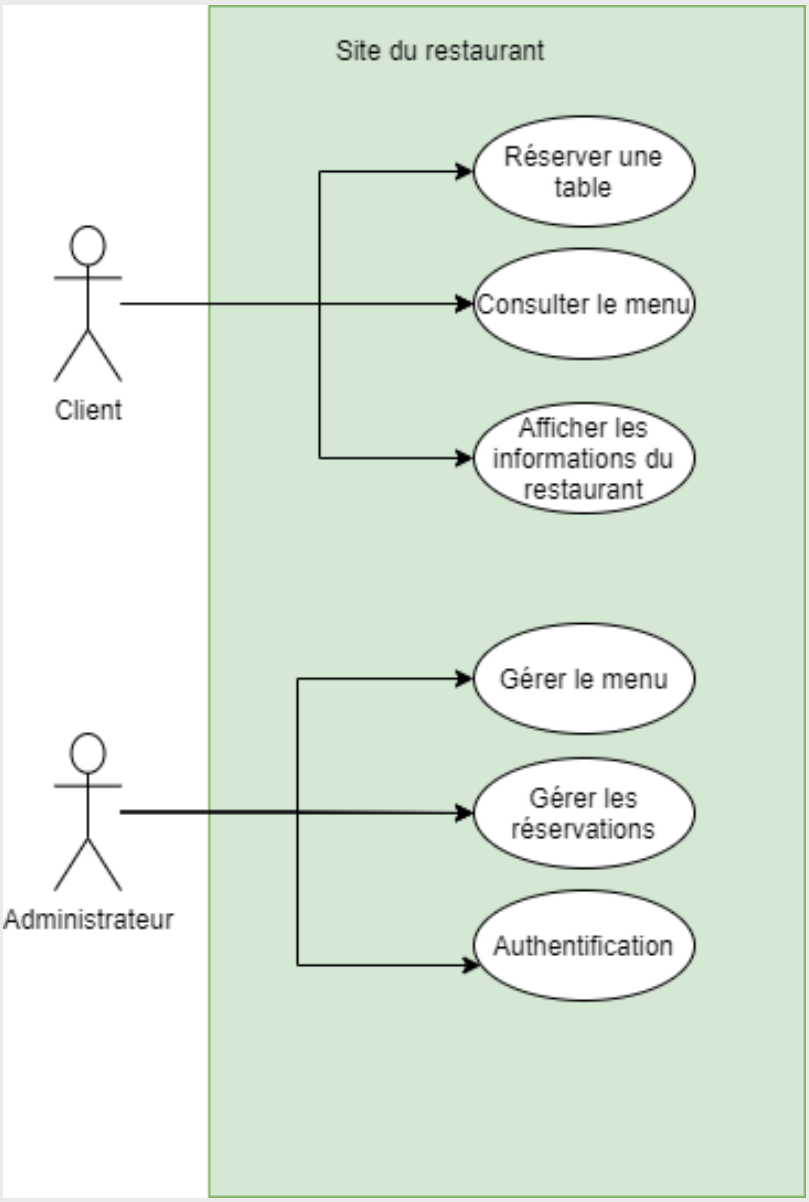
Exemple **Diagramme de cas d'utilisation**

Nous devons réaliser le diagramme de cas d'utilisation de notre application. Le diagramme d'utilisation existe pour représenter toutes les interactions possibles pour tous les types d'utilisateurs. Et ce, de façon exhaustive.

Dans notre application, nous avons deux types d'utilisateurs : l'utilisateur connecté (administrateur) et l'utilisateur déconnecté (le client).

Le client pourra donc réserver une table et consulter toutes les informations du restaurant (menu, horaires, etc.). L'administrateur quant à lui, pourra gérer les menus, et consulter ou modifier toutes les réservations.

De ces informations, nous pouvons générer le diagramme suivant :



DiagrammeCasUtilisation

Dans ce diagramme, nous avons deux acteurs principaux : le « *Client* » et l'« *Administrateur* ». Le « *Client* » représente les utilisateurs du site web qui visitent le restaurant en ligne, tandis que l'« *Administrateur* » est responsable de la gestion du contenu et des fonctionnalités du site.

Les cas d'utilisation sont représentés sous forme de boîtes en ellipse et décrivent les actions que les acteurs peuvent effectuer. Voici une explication de chaque cas d'utilisation :

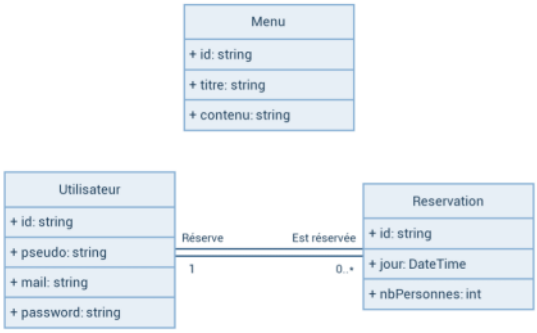
- « *Réserver une table* » : le client peut utiliser cette fonctionnalité pour réserver une table en ligne.
- « *Consulter le menu* » : le client peut consulter le menu du restaurant en ligne pour voir les plats proposés.
- « *Afficher les informations du restaurant* » : le client peut accéder aux informations générales sur le restaurant telles que les horaires d'ouverture, l'emplacement et les spécialités.
- « *Gérer le menu* » : l'administrateur peut gérer le contenu du menu en ajoutant, modifiant ou supprimant des plats.

- « *Gérer les réservations* » : l'administrateur peut gérer les réservations effectuées par les clients en confirmant ou en annulant les réservations.
- « *Authentification* » : l'administrateur doit s'authentifier pour accéder aux fonctionnalités réservées à l'administration du site.

Les flèches reliant les acteurs aux cas d'utilisation représentent les interactions entre eux. Par exemple, le « *Client* » peut effectuer les actions de « *Réserver une table* », « *Consulter le menu* » et « *Afficher les informations du restaurant* ». De même, l'« *Administrateur* » peut réaliser les actions de « *Gérer le menu* », « *Gérer les réservations* » et « *Authentification* ».

Diagramme de classes

Nous devons maintenant générer le diagramme de classe de notre application. Le diagramme de classe est là pour matérialiser toutes les données que nous devons gérer dans notre application. Nous voulons pouvoir gérer les menus de notre restaurant, les utilisateurs pouvant se connecter à notre site, ainsi que toutes les réservations. De plus, les réservations sont faites par un utilisateur. Grâce à ces informations nous pouvons construire le diagramme suivant :



Ce diagramme de classes représente trois classes principales : Utilisateur, Menu et Reservation. Voici une description des attributs de chaque classe :

- Utilisateur :
  - id : identifiant de l'utilisateur (de type string)
  - pseudo : nom d'utilisateur (de type string)
  - mail : adresse e-mail de l'utilisateur (de type string)
  - password : mot de passe de l'utilisateur (de type string)
- Menu :
  - id : identifiant du menu (de type string)
  - titre : titre du menu (de type string)
  - contenu : contenu du menu (de type string)
- Réservation :
  - id : identifiant de la réservation (de type string)
  - jour : date et heure de la réservation (de type DateTime)
  - nbPersonnes : nombre de personnes pour la réservation (de type int)

De plus, le diagramme montre une relation « *Réserve* » entre Utilisateur et Reservation, indiquant qu'un Utilisateur peut effectuer plusieurs réservations.

Ici, toutes les propriétés sont précédées d'un « + ». Ce qui signifie que la propriété est publique. Elles sont accessibles à partir de n'importe quelle classe ou objet. Cela signifie que d'autres classes peuvent lire et modifier ces propriétés directement.

Définition **Cardinalités**

La cardinalité est généralement représentée près de l'association entre deux classes dans un diagramme de classes. Voici quelques exemples courants de notation de cardinalité en UML :

- Cardinalité "1" : indique qu'il y a exactement une occurrence de l'entité associée. Cela peut signifier qu'une classe est associée à une seule instance d'une autre classe.
- Cardinalité "0..1" : indique qu'il peut y avoir zéro ou une occurrence de l'entité associée. Cela signifie qu'une classe peut être optionnellement associée à une instance d'une autre classe.
- Cardinalité "0.." ou "0..*" : indique qu'il peut y avoir zéro ou plusieurs occurrences de l'entité associée. Cela signifie qu'une classe peut être associée à zéro ou plusieurs instances d'une autre classe.
- Cardinalité "1..*" : indique qu'il doit y avoir au moins une occurrence de l'entité associée. Cela signifie qu'une classe est associée à une ou plusieurs instances d'une autre classe.

La notation de la cardinalité peut varier légèrement selon les conventions utilisées, mais les chiffres ou symboles spécifiques sont généralement placés près de l'association pour indiquer la multiplicité de la relation.

La cardinalité est importante pour définir les relations et les contraintes entre les entités dans un système logiciel. Elle permet de spécifier combien d'instances d'une classe peuvent être associées à d'autres instances d'une autre classe, ce qui aide à modéliser les interactions et les dépendances entre les entités dans un système.

IV. Essentiel

Nous le comprenons désormais, avant de commencer le développement d'un logiciel, nous avons besoin d'une étape de conception de l'application.

Depuis l'écoute du besoin client, nous allons pouvoir créer un cahier des charges, document contenant la liste des demandes de l'utilisateur. Ce document, nous allons ensuite le traduire en liste de fonctionnalités : il s'agit des spécifications fonctionnelles. Ces dernières seront utiles pour concevoir l'architecture logiciel de l'application. Le document contenant toutes les informations sur l'architecture technique se nomme « *spécifications techniques* ».

Toutes ces étapes peuvent être schématisés par des diagrammes. Pour cela, vous pouvez utiliser le langage UML. Vous pourrez ainsi réaliser des diagrammes compréhensibles par n'importe quel développeur. Ce langage défini 14 diagrammes regroupés dans deux catégories : diagrammes de comportement (spécifications fonctionnelles), et les diagrammes de structure (spécifications techniques).

Nous avons vu comment utiliser quelques-uns de ces diagrammes, et en quoi ils sont fondamentaux pour réaliser des outils utiles.

V. Auto-évaluation

A. Exercice

Vous travaillez pour une entreprise de développement de logiciels et vous êtes en train d'élaborer un nouveau système de gestion de bibliothèque. Votre mission est de créer une modélisation conceptuelle de données à l'aide d'UML pour illustrer les différents acteurs du système et leurs interactions.

Vous avez le cahier des charges suivants :

Cahier des charges : MyBibliotheque

1. Objectif du projet

L'objectif de ce projet est de développer un système de gestion de bibliothèque informatisé qui permettra à la bibliothèque de gérer efficacement ses opérations quotidiennes. Ce système doit offrir des fonctionnalités pour les administrateurs de la bibliothèque ainsi que pour les clients.

2. Public cible

- Administrateurs de la bibliothèque
- Clients de la bibliothèque

3. Fonctionnalités principales

Pour les **administrateurs** :

- Gérer les livres : les administrateurs doivent être capables de créer, modifier et supprimer des livres.
- Gérer les comptes des utilisateurs : les administrateurs doivent être capables de créer, modifier et supprimer des comptes d'utilisateurs.
- Rechercher un livre : les administrateurs doivent être capables de rechercher un livre par titre, auteur, genre, etc.

Pour les **clients** :

- Rechercher un livre : les clients doivent être capables de rechercher un livre par titre, auteur, genre, etc.
- Emprunter un livre : les clients doivent être capables d'emprunter un livre disponible dans le système.
- Retourner un livre : les clients doivent être capables de retourner un livre qu'ils ont précédemment emprunté.
- Consulter son historique d'emprunts : les clients doivent être capables de consulter leur historique d'emprunts.

4. Technologie et environnement de développement

Le système sera développé en utilisant react côté front, et ASP.NET CORE côté back. Les développeurs utiliseront un logiciel de modélisation UML pour créer des modèles conceptuels de données.

5. Contraintes du projet

- Le système doit être facile à utiliser et intuitif pour tous les utilisateurs.
- Le système doit garantir la confidentialité et la sécurité des informations des utilisateurs.
- Le système doit être capable de gérer une grande quantité de données sans affecter ses performances.
- Le système doit être développé dans le respect du budget et du calendrier convenus.

6. Livrables

- Diagrammes UML (cas d'utilisation, diagrammes de classes, etc.)
- Code source du système
- Documentation du système (manuel d'utilisation, documentation technique)
- Rapport de test du système

Question 1

[solution n°1 p.15]

Modélisez les cas d'utilisation pour l'Administrateur.

Question 2

[solution n°2 p.15]

Modélisez les cas d'utilisation pour un Client Connecté.

B. Test

Exercice 1 : Quiz

[solution n°3 p.15]

Question 1

Quelles sont les étapes de la conception d'une application ?

- ○ Récupération du besoin client, spécifications fonctionnelles, spécifications techniques, développement
- ○ Récupération du besoin client, développement, spécifications fonctionnelles, spécifications techniques
- ○ Développement, spécifications fonctionnelles, spécifications techniques, récupération du besoin client

Question 2

Qu'est-ce qu'une spécification fonctionnelle ?

- ○ Un document détaillant les fonctionnalités et les comportements attendus d'un produit informatique
- ○ Un document décrivant les exigences et les caractéristiques techniques d'un produit ou d'un système
- ○ Un document définissant les contraintes budgétaires et les délais de développement d'un projet

Question 3

Qu'est-ce qu'une spécification technique ?

- ○ Un document détaillant les fonctionnalités et les comportements attendus d'un produit informatique
- ○ Un document décrivant les exigences et les caractéristiques techniques d'un produit ou d'un système
- ○ Un document définissant les contraintes budgétaires et les délais de développement d'un projet

Question 4

Qu'est-ce que l'UML ?

- ○ Un langage graphique utilisé pour représenter visuellement les systèmes logiciels
- ○ Une méthode de gestion de projet agile
- ○ Un logiciel de développement intégré pour programmer des applications

Question 5

Quels sont les types de diagrammes en UML ?

- ○ Diagrammes de structure et diagrammes de comportement
- ○ Diagrammes de planification et diagrammes d'exécution
- ○ Diagrammes de conception et diagrammes de déploiement

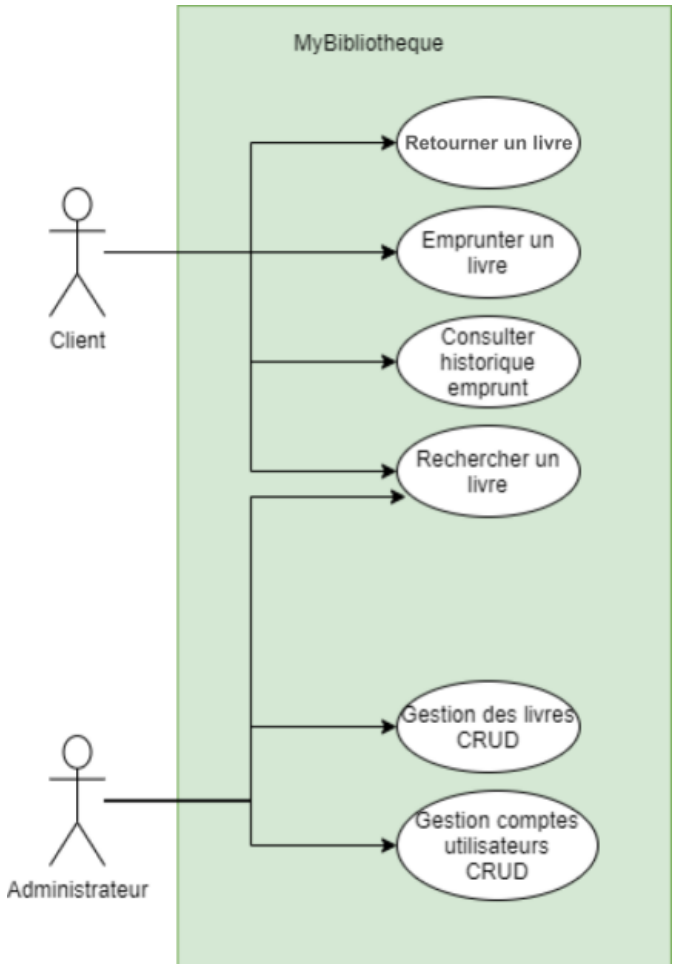
Solutions des exercices

p. 13 **Solution n°1**

Un diagramme de cas d'utilisation pour l'administrateur pourrait contenir deux cas d'utilisation. Le premier serait la gestion des livres : CRUD (Create, Read, Update, Delete) sur les livres de la bibliothèque. Le deuxième pourrait être la gestion des comptes des utilisateurs : CRUD sur les utilisateurs.

p. 13 **Solution n°2**

Un diagramme de cas d'utilisation pour un client connecté pourrait contenir quatre cas d'utilisation suivants. Le premier cas pourrait être « *Rechercher un livre* » : le client peut rechercher un livre par titre, auteur, genre, etc. Le système fournit les résultats correspondants. Le deuxième peut être « *Emprunter un livre* » : le client peut sélectionner un livre à emprunter. Le système met à jour le statut du livre et l'historique d'emprunts du client. Le troisième est « *Retourner un livre* » : le client peut retourner un livre précédemment emprunté. Le système met à jour le statut du livre et l'historique d'emprunts du client. Et enfin, le dernier peut être « *Consulter son historique d'emprunts* » : le client peut consulter son historique d'emprunts. Le système affiche une liste de tous les livres que le client a empruntés et la date de chaque emprunt.



CasFinal

Exercice p. 13 **Solution n°3**

Question 1

Quelles sont les étapes de la conception d'une application ?

- ◉ Récupération du besoin client, spécifications fonctionnelles, spécifications techniques, développement
- ○ Récupération du besoin client, développement, spécifications fonctionnelles, spécifications techniques
- ○ Développement, spécifications fonctionnelles, spécifications techniques, récupération du besoin client

Q Les étapes de la conception d'une application sont dans l'ordre : récupération du besoin client, spécifications fonctionnelles, spécifications techniques, développement. Ces étapes permettent de définir le périmètre fonctionnel de l'application, de spécifier les règles à respecter et les contraintes techniques, avant de passer à la phase de développement.

Question 2

Qu'est-ce qu'une spécification fonctionnelle ?

- ◉ Un document détaillant les fonctionnalités et les comportements attendus d'un produit informatique
- ○ Un document décrivant les exigences et les caractéristiques techniques d'un produit ou d'un système
- ○ Un document définissant les contraintes budgétaires et les délais de développement d'un projet

Q Une spécification fonctionnelle est un document détaillant les fonctionnalités et les comportements attendus d'un produit, d'un système logiciel ou d'une application. Elle décrit les différentes actions, tâches ou opérations que le système doit être capable d'effectuer pour répondre aux besoins des utilisateurs.

Question 3

Qu'est-ce qu'une spécification technique ?

- ○ Un document détaillant les fonctionnalités et les comportements attendus d'un produit informatique
- ◉ Un document décrivant les exigences et les caractéristiques techniques d'un produit ou d'un système
- ○ Un document définissant les contraintes budgétaires et les délais de développement d'un projet

Q Une spécification technique est un document qui décrit de manière détaillée les exigences et les caractéristiques techniques d'un produit ou d'un système. Elle fournit des informations spécifiques sur les composants, les fonctionnalités, les performances, les normes et les contraintes techniques nécessaires à la conception et au développement du produit ou du système.

Question 4

Qu'est-ce que l'UML ?

- ◉ Un langage graphique utilisé pour représenter visuellement les systèmes logiciels
- ○ Une méthode de gestion de projet agile
- ○ Un logiciel de développement intégré pour programmer des applications

Q UML est un langage graphique utilisé pour représenter visuellement les systèmes logiciels. Il propose des symboles et des schémas standardisés qui aident les développeurs à comprendre et à communiquer sur la structure, le comportement et les interactions d'un logiciel.

Question 5

Quels sont les types de diagrammes en UML ?

- (●) Diagrammes de structure et diagrammes de comportement
- ( ) Diagrammes de planification et diagrammes d'exécution
- ( ) Diagrammes de conception et diagrammes de déploiement

Les types de diagrammes en UML sont les diagrammes de structure et les diagrammes de comportement. Les diagrammes de structure permettent de définir les différents composants de l'application, tandis que les diagrammes de comportement décrivent les fonctionnalités et les interactions dans une application.