Bouzidi Abdessalem
Larbi Nidal
Sene Pape Adama

My Kitchen : System Analysis

Sommaire

I. Introduction :

A) Rappel des Objectifs de l'Application MyKitchen :

Le projet MyKitchen incarne une vision innovante dans le domaine de la conception et du réaménagement des espaces de cuisine. Visant à développer une application mobile de pointe, MyKitchen ambitionne de transformer radicalement la manière dont les utilisateurs envisagent la planification de leurs cuisines. Cette application se propose comme un outil virtuel permettant aux utilisateurs de fournir le plan de leur cuisine existante et de réorganiser à leur guise les éléments constitutifs tels que les appareils électroménagers, les meubles, et les accessoires. Une des caractéristiques remarquables de MyKitchen est sa capacité à calculer instantanément le coût total des modifications apportées, offrant ainsi une perspective complète sur le budget nécessaire pour concrétiser le projet d'aménagement envisagé.

1. Les objectifs principaux de MyKitchen se déclinent comme suit :

a) **Facilité d'Usage :** Offrir une plateforme conviviale permettant aux utilisateurs de concevoir et d'agencer leur cuisine en fonction de leurs préférences personnelles et de leurs exigences spécifiques, sans nécessiter de compétences techniques particulières.

b) **Fonctionnalités Avancées :** Intégrer des outils de simulation 2D/3D sophistiqués pour permettre une visualisation immersive et réaliste du projet final, aidant ainsi les utilisateurs à se projeter dans leur futur espace de cuisine avant même sa réalisation.

c) **Expérience Utilisateur Optimale :** S'assurer que l'interface de l'application soit intuitive et enrichie de fonctionnalités de personnalisation, pour une navigation aisée et une expérience utilisateur enrichissante.

d) **Transparence Budgétaire :** Fournir un aperçu clair et détaillé des coûts associés à chaque élément de la cuisine, permettant ainsi une gestion budgétaire éclairée et prévenant toute surprise financière.

Sur le plan technologique, le développement de MyKitchen s'appuie sur l'utilisation de technologies avancées comme la modélisation 2D/3D et des interfaces utilisateurs intuitives, le tout développé avec des langages de programmation adaptés au mobile. Cette approche assure non seulement une grande qualité visuelle mais également une facilité d'interaction pour l'utilisateur.

Commercialement, MyKitchen répond à une demande croissante pour des outils numériques facilitant la conception et l'agencement des espaces de vie, particulièrement des cuisines. À une époque où la personnalisation et la décoration intérieure gagnent en popularité, MyKitchen se positionne comme une solution innovante et pratique, répondant aux aspirations des consommateurs désireux de remodeler leurs cuisines selon leurs goûts uniques, tout en bénéficiant d'une visualisation précise et d'une estimation budgétaire fiable de leur projet.

B) Les contraintes et les limites de l'application MyKitchen

1. Précision sur les fonctionnalités et aspects non présents dans le cahier des charges

Le Cahier des Charges (CDC) de MyKitchen établit des directives claires pour le développement de l'application, mais certaines fonctionnalités ou aspects pourraient ne pas être présents en raison des contraintes identifiées. Parmi celles-ci, nous pouvons citer :

a) **Compatibilité Limitée :** L'application est conçue pour être compatible avec les systèmes d'exploitation iOS et Android sur tablette, ce qui pourrait exclure d'autres plateformes ou appareils moins courants.

b) **Exigences Techniques :** La nécessité d'avoir au moins 3 Gde RAM sur les tablettes pour une expérience utilisateur fluide pourrait limiter l'accessibilité à l'application pour les utilisateurs ayant des appareils plus anciens ou moins performants.

c) **Stockage et Sécurité des Données :** La contrainte d'utiliser une base de données sécurisée et conforme aux réglementations (comme le GDPR) peut limiter la flexibilité dans le choix des solutions de stockage de données, et nécessiter des efforts supplémentaires en termes de sécurisation et de conformité.

2. Complexité du Projet

Les principales contraintes et problèmes potentiels associés au projet MyKitchen comprennent :

a) **Contraintes techniques :** La modélisation 2D/3D, la compatibilité inter-plateformes, et l'intégration sécurisée des bases de données représentent des défis techniques significatifs, nécessitant une expertise spécialisée et potentiellement augmentant les coûts de développement et les délais de mise en œuvre.

b) **Contraintes Economiques et Environnementales :** Avec un budget limité alloué à la formation (50 euros), il pourrait y avoir des limites dans l'acquisition de compétences ou de connaissances spécifiques nécessaires au projet. De plus, bien que l'application promeuve des pratiques environnementales en réduisant la demande de matériaux physiques, la mise en œuvre effective de ces pratiques et leur impact sur l'expérience utilisateur restent à voir.

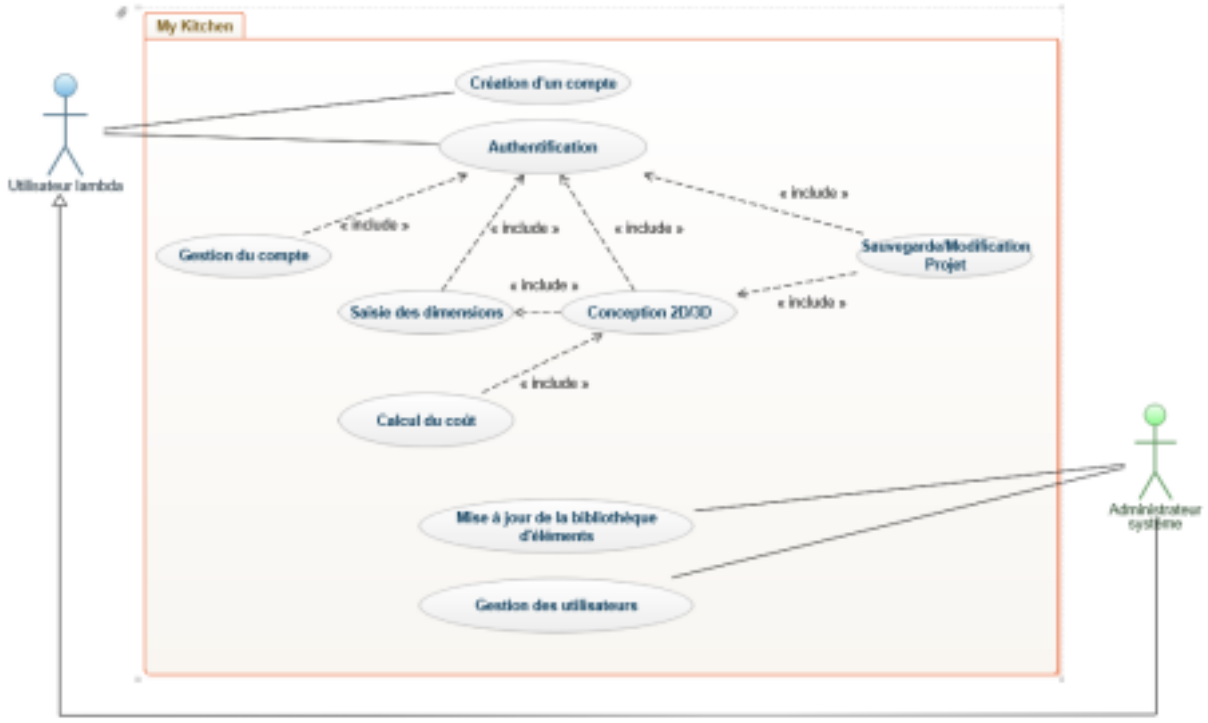
c) **Sécurité et Confidentialité :** Assurer la sécurité des intégrations avec des services tiers ou des API, ainsi que la protection complète des données personnelles des utilisateurs, est essentiel mais peut s'avérer complexe, en particulier dans le respect de réglementations strictes comme le GDPR.

Ces contraintes soulignent l'importance d'une planification minutieuse, d'une gestion des risques efficace, et d'une allocation de ressources adéquate pour surmonter les défis et réaliser avec succès le projet MyKitchen.

Ayant exploré les dimensions de l'application MyKitchen, depuis ses objectifs et fonctionnalités jusqu'aux contraintes et limites, il est désormais pertinent de se pencher sur les acteurs qui animent le système. Ces acteurs, incarnés par les classes d'utilisateurs et d'administrateurs, jouent un rôle crucial dans la dynamique et l'efficacité de l'application. La classe "User" représente les utilisateurs finaux de l'application, avec des fonctionnalités leur permettant de créer, gérer et modifier des projets de cuisine. À l'autre extrémité, la classe "Administrator" incarne les administrateurs du système, dotés de privilèges étendus pour superviser et gérer les aspects critiques de l'application, tels que la gestion des utilisateurs et des projets. Ces acteurs et leurs interactions définissent non seulement la structure opérationnelle de MyKitchen mais aussi son efficacité dans la réalisation des ambitions et des attentes des utilisateurs.

II. The actors of the system (Use Case Diagram) :

A. Use Case Diagram of My Kitchen :



B. Brief description of each case :

Description textuelle du cas d'utilisation : Création d'un Compte

- **Nom :** Création d'un Compte.
- **Objectif :** Permettre à un nouvel utilisateur de créer un compte personnel pour accéder et utiliser l'application MyKitchen.
- **Acteur principal :** Nouvel Utilisateur
- **Statut :** Principal
- **Étapes principales :**
    1. L'utilisateur sélectionne l'option de création de compte sur l'interface de l'application.
    2. L'utilisateur est invité à fournir des informations personnelles, notamment un nom, une adresse e-mail et un mot de passe.
    3. L'application vérifie que l'adresse e-mail n'est pas déjà utilisée et que le mot de passe répond aux critères de sécurité.
    4. L'utilisateur confirme les informations et soumet le formulaire d'inscription. 5. L'application enregistre le nouvel utilisateur dans la base de données et crée le compte.
    6. L'utilisateur reçoit une confirmation que son compte a été créé avec succès.
- **Extensions :**
    - **Validation de l'adresse e-mail** :

▪ **Point de rattachement de l'extension :** Après l'étape 3
▪ **Étapes principales de l'extension :**
1. Si l'adresse e-mail est déjà utilisée, l'application informe l'utilisateur et invite à essayer une autre adresse.
2. Si le mot de passe ne répond pas aux critères, l'application affiche les exigences de sécurité et demande à l'utilisateur de saisir un nouveau mot de passe.

Description textuelle du cas d'utilisation : Authentification

• **Nom :** Authentification
• **Objectif :** Permettre à un utilisateur enregistré de s'authentifier afin d'accéder à son compte personnel et d'utiliser les fonctionnalités de l'application MyKitchen. • **Acteur principal :** Utilisateur Enregistré
• **Acteurs secondaires :** Aucun
• **Statut :** Principal
• **Étapes principales :**
1. L'utilisateur lance l'application MyKitchen et sélectionne l'option de connexion. 2. L'utilisateur est invité à saisir son adresse e-mail et son mot de passe. 3. L'application vérifie les informations d'identification auprès de la base de données.
4. Si les informations sont correctes, l'application accorde l'accès à l'utilisateur et charge son profil.
5. L'utilisateur est maintenant connecté et peut utiliser pleinement l'application.

• **Extensions :**
o **Récupération du mot de passe oublié** :
▪ **Point de rattachement de l'extension :** Après l'étape 2, si l'utilisateur ne se souvient pas de son mot de passe.
▪ **Étapes principales de l'extension :**
1. L'utilisateur sélectionne l'option « Mot de passe oublié ».
2. L'application demande à l'utilisateur de saisir son adresse e mail.
3. L'utilisateur reçoit un e-mail avec des instructions pour réinitialiser son mot de passe.
o **Échec de l'authentification** :

- **Point de rattachement de l'extension :** Après l'étape 3, si les informations d'identification sont incorrectes.
- **Étapes principales de l'extension :**
  1. L'application affiche un message d'erreur indiquant que l'adresse e-mail ou le mot de passe est incorrect.
  2. L'utilisateur a la possibilité de réessayer de saisir ses informations d'identification ou de récupérer son mot de passe.

Description textuelle du cas d'utilisation : Gestion du Compte.

- **Nom :** Gestion du Compte
- **Objectif :** Permettre à l'utilisateur authentifié de mettre à jour ses informations personnelles, de changer son mot de passe et de consulter l'historique de ses projets de conception de cuisine.
- **Acteur principal :** Utilisateur Authentifié
- **Acteurs secondaires :** Aucun
- **Statut :** Principal
- **Étapes principales :**
  1. L'utilisateur authentifié accède à la section de gestion de compte de l'application MyKitchen.
  2. L'utilisateur peut choisir de mettre à jour ses informations personnelles, de changer son mot de passe ou de consulter l'historique de ses projets.
  3. Pour modifier les informations, l'utilisateur doit entrer les nouvelles données et les soumettre.
  4. Pour changer le mot de passe, l'utilisateur doit entrer l'ancien mot de passe, le nouveau et confirmer le nouveau.
  5. Pour consulter l'historique, l'utilisateur peut simplement visualiser les projets passés et leur statut.
  6. L'application enregistre les modifications apportées par l'utilisateur et met à jour la base de données.

- **Inclusion :**
  - **Authentification** « includes » **Gestion du Compte**
    - Avant que l'utilisateur puisse accéder à la gestion de son compte, il doit être authentifié.
    - Si l'utilisateur tente d'accéder à la gestion de compte sans être authentifié, il sera redirigé vers le processus d'authentification.
- **Extensions :**
  - **Mise à jour échouée** :

▪ **Point de rattachement de l'extension :** Après l'étape 3 ou 4, si les modifications ne sont pas enregistrées correctement.

▪ **Étapes principales de l'extension :**

1. L'application affiche un message d'erreur indiquant que la mise à jour a échoué.
2. L'utilisateur peut réessayer d'entrer les informations et de les soumettre à nouveau.

Description textuelle du cas d'utilisation : Saisie des Dimensions.

• **Nom :** Saisie des Dimensions

• **Objectif :** Permettre à l'utilisateur de saisir les dimensions précises de son espace de cuisine pour assurer que la conception s'adapte parfaitement à l'environnement réel. • **Acteur principal :** Utilisateur Authentifié

• **Acteurs secondaires :** Aucun

• **Statut :** Principal

• **Étapes principales :**

1. L'utilisateur authentifié sélectionne l'option pour commencer un nouveau projet de conception de cuisine.
2. L'application invite l'utilisateur à saisir les dimensions de l'espace de cuisine, y compris la longueur, la largeur, et la hauteur.
3. L'utilisateur entre les dimensions requises dans les champs prévus à cet effet. 4. L'application valide les dimensions saisies et les applique au modèle de conception de la cuisine.
5. Les dimensions sont enregistrées avec le projet, permettant à l'utilisateur de procéder avec la conception.

• **Inclusion :**

o **Authentification** « includes » **Saisie des Dimensions**

▪ L'utilisateur doit être authentifié avant de pouvoir saisir les dimensions de son espace de cuisine.

▪ Si l'utilisateur n'est pas authentifié, il sera redirigé vers le processus d'authentification avant de pouvoir accéder à cette fonctionnalité.

• **Extensions :**

o **Dimensions incorrectes** :

▪ **Point de rattachement de l'extension :** Après l'étape 4, si les dimensions sont hors des limites acceptables ou mal formatées.

▪ **Étapes principales de l'extension :**

1. L'application affiche un message d'erreur indiquant que les dimensions sont incorrectes.
2. L'utilisateur est invité à les corriger et à les ressaisir.

Description textuelle du cas d'utilisation : Conception 3D/2D :

- **Nom :** Conception 3D/2D
- **Objectif :** Permettre à l'utilisateur de créer et de visualiser un modèle 3D/2D de son espace de cuisine, basé sur les dimensions réelles saisies.
- **Acteur principal :** Utilisateur Authentifié
- **Acteurs secondaires :** Aucun
- **Statut :** Principal
- **Étapes principales :**
    1. L'utilisateur authentifié et ayant saisi les dimensions choisit de commencer la conception en 3D/2D de la cuisine.
    2. L'application présente une interface de modélisation où l'utilisateur peut sélectionner et placer des éléments de cuisine.
    3. L'utilisateur utilise les outils fournis pour ajouter, retirer ou modifier les éléments de la cuisine dans la maquette.
    4. Une fois satisfait, l'utilisateur peut enregistrer le modèle de conception ou continuer à le modifier.
- **Inclusion :**
    - **Authentification** « includes » **Conception 3D/2D**
        - L'utilisateur doit être authentifié pour accéder aux outils de conception 3D/2D.
    - **Saisie des Dimensions** « includes » **Conception 3D/2D**
        - Les dimensions de l'espace de cuisine doivent être saisies avant de commencer la conception en 3D/2D.
- **Extensions :**
    - **Ajustement des dimensions** :
        - **Point de rattachement de l'extension :** Pendant l'étape 3, si l'utilisateur souhaite ajuster les dimensions originales.
        - **Étapes principales de l'extension :**
            1. L'utilisateur sélectionne l'option d'ajustement des dimensions.
            2. L'application permet à l'utilisateur de modifier les dimensions et met à jour la maquette en conséquence.

Description textuelle du cas d'utilisation : Sauvegarde

/ Modification de Projet

- **Nom :** Sauvegarde / Modification de Projet
- **Objectif :** Permettre à l'utilisateur de sauvegarder son projet de conception de cuisine et de le modifier ultérieurement.
- **Acteur principal :** Utilisateur Authentifié
- **Acteurs secondaires :** Aucun
- **Statut :** Principal
- **Étapes principales :**
    1. L'utilisateur authentifié accède à son projet de conception de cuisine en 3D/2D.
    2. L'utilisateur peut choisir de sauvegarder l'état actuel du projet pour une consultation ou une modification future.
    3. Si l'utilisateur choisit de modifier le projet, l'application permet l'accès à l'interface de modélisation où des changements peuvent être effectués. 4. Après la modification, l'utilisateur peut sauvegarder les nouveaux changements ou annuler les modifications récentes.
    5. L'application enregistre la version la plus récente du projet dans la base de données de l'utilisateur.
    6. Si l'utilisateur décide de supprimer un projet, l'application lui permet de sélectionner le projet à supprimer et confirme son choix avant de procéder à la suppression.
- **Inclusion :**
    - **Authentification** « includes » **Sauvegarde / Modification de Projet**
        - L'utilisateur doit être authentifié pour sauvegarder ou modifier un projet.
    - **Conception 3D/2D** « includes » **Sauvegarde / Modification de Projet**
        - L'utilisateur doit avoir accédé à la conception 3D/2D avant de sauvegarder ou de modifier le projet.

- **Extensions :**

    - **Reprise d'un projet sauvegardé** :

        - **Point de rattachement de l'extension :** Après l'étape 1, si l'utilisateur souhaite reprendre un projet sauvegardé.
        - **Étapes principales de l'extension :**
            1. L'utilisateur sélectionne un projet existant à partir de

l'historique de ses projets sauvegardés.
2. L'application charge le projet pour que l'utilisateur puisse le consulter ou le modifier.

o **Suppression d'un projet :**

- **Point de rattachement de l'extension :** Après l'étape 1 ou à tout moment où l'utilisateur souhaite supprimer un projet.
- **Étapes principales de l'extension :**
  1. L'utilisateur sélectionne l'option de supprimer un projet.
  2. L'application affiche une liste des projets existants et permet à l'utilisateur de choisir celui qu'il souhaite supprimer.
  3. L'utilisateur confirme son choix de suppression.
  4. L'application supprime le projet sélectionné de la base de données de l'utilisateur.

Description textuelle du cas d'utilisation : Mise à jour de la bibliothèque

• **Nom :** Mise à jour de la bibliothèque
• **Objectif :** Assurer que la bibliothèque d'éléments de cuisine disponibles pour la conception dans l'application MyKitchen soit à jour, précise et reflète les derniers produits et tendances.
• **Acteur principal :** Administrateur
• **Acteurs secondaires :** Aucun (cette tâche se déroule en back-end sans interaction directe avec les utilisateurs finaux)
• **Statut :** Principal
• **Étapes principales :**
1. L'administrateur se connecte au système de gestion de contenu back-end de l'application MyKitchen.
2. L'administrateur accède à la section de la base de données où la bibliothèque d'éléments est stockée.
3. Il procède à l'ajout de nouveaux éléments, à la mise à jour des éléments existants ou à la suppression d'éléments obsolètes.
4. L'administrateur valide les modifications pour s'assurer de leur exactitude. 5. Les changements sont enregistrés dans la base de données et deviennent disponibles pour les utilisateurs lors de la conception de leur cuisine. • **Extensions :**

- **Validation des informations produit** :
  - **Point de rattachement de l'extension :** Après l'étape 3, lorsque de nouvelles informations produit sont saisies ou modifiées.
  - **Étapes principales de l'extension :**
    1. Un système de validation vérifie les nouvelles informations pour s'assurer qu'elles sont complètes et conformes aux standards de l'application.
    2. Si des erreurs sont détectées ou si des informations sont manquantes, l'administrateur est alerté et doit corriger les données avant de les enregistrer.

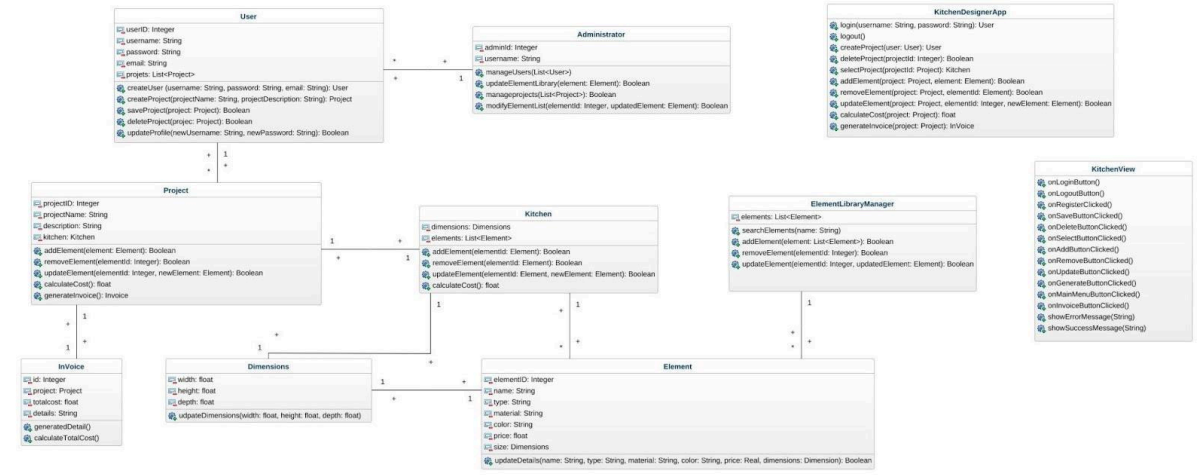
Description textuelle du cas d'utilisation : Gestion des Utilisateurs :

- **Nom :** Gestion des Utilisateurs
- **Objectif :** Permettre à l'administrateur de gérer les comptes des utilisateurs, y compris la création, la mise à jour, la désactivation et la suppression de comptes.
- **Acteur principal :** Administrateur
- **Acteurs secondaires :** Aucun
- **Statut :** Principal
- **Étapes principales :**
  1. L'administrateur accède à la console de gestion des utilisateurs de l'application MyKitchen.
  2. L'administrateur peut effectuer une ou plusieurs des actions suivantes :
     - Créer de nouveaux comptes d'utilisateurs en fournissant les informations nécessaires telles que nom, e-mail, et rôle.
     - Mettre à jour les informations des comptes d'utilisateurs existants, comme modifier les rôles d'accès ou mettre à jour les coordonnées.
     - Désactiver des comptes d'utilisateurs pour restreindre temporairement l'accès à l'application.
     - Supprimer des comptes d'utilisateurs pour retirer définitivement l'accès et les données associées.
  3. L'application valide les actions de l'administrateur et met à jour la base de données des utilisateurs en conséquence.
  4. L'administrateur reçoit une confirmation que les actions de gestion des utilisateurs ont été exécutées avec succès.
- **Cas d'exception :**
  - Si l'administrateur tente de supprimer ou de modifier un compte qui

n'existe pas ou qui a déjà été supprimé, l'application affiche un message d'erreur indiquant que l'action ne peut pas être complétée.

III. The data necessary understand the system (Diagramme de classes) :

A. Diagramme de Classes :



B. Descriptions des objets du Système :

Classe User :

La classe **User** représente un utilisateur au sein de l'application MyKitchen.

Attributs :

- **userID (Integer)** : Identifiant unique pour chaque utilisateur.

- **username (String)** : Nom d'utilisateur utilisé pour se connecter à l'application.
- **password (String)** : Mot de passe de l'utilisateur pour sécuriser l'accès au compte.
- **email (String)** : Adresse e-mail de l'utilisateur, probablement utilisée pour la communication et la récupération du compte.
- **projets (List<Project>)** : Collection de projets associés à l'utilisateur.

Méthodes :

- **createUser(username: String, password: String, email: String): User :** Crée et retourne une nouvelle instance de User avec les informations fournies.
- **createProject(projectName: String, projectDescription: String): Project :** Crée un nouveau projet avec un nom et une description, et le lie à l'utilisateur.
- **saveProject(project: Project): Boolean :** Enregistre le projet spécifié dans la liste des projets de l'utilisateur. Retourne true si l'opération est réussie.
- **deleteProject(projectId: Integer): Boolean :** Supprime le projet correspondant à l'ID spécifié de la liste des projets de l'utilisateur. Retourne true si la suppression est réussie.
- **updateProfile(newUsername: String, newPassword: String): Boolean** : Met à jour le profil de l'utilisateur avec un nouveau nom d'utilisateur et/ou un nouveau mot de passe. Retourne true si la mise à jour est réussie.

Classe Project :

La classe **Project** représente un projet au sein de l'application MyKitchen.

Attributs :

- **projectID (Integer**) : Identifiant unique du projet.
- **projectName (String)** : Nom donné au projet.
- **description (String)** : Description textuelle du projet.
- **kitchen (Kitchen)** : Une instance de la classe Kitchen qui représente la cuisine associée à ce projet.

Méthodes :

- **addElement(element: Element): Boolean** : Ajoute un élément à la cuisine du projet. Retourne true si l'ajout est réussi.
- **removeElement(elementId: Integer): Boolean** : Supprime un élément de la cuisine en se basant sur son identifiant. Retourne true si la suppression est réussie.
- **updateElement(elementId: Integer, newElement: Element): Boolean** : Met à jour les informations d'un élément existant dans la cuisine. Retourne true si la mise à jour est réussie.
- **calculateCost(): Real** : Calcule et retourne le coût total du projet en se basant sur les éléments inclus dans la cuisine.
- **generateInvoice(): Invoice** : Génère et retourne une facture pour le projet, probablement en détaillant les coûts des différents éléments et le coût total.

Classe InVoice :

La classe **InVoice** est conçue pour représenter une facture liée à un projet de conception.

Attributs :

- **id (Integer) :** Identifiant unique pour chaque facture.
- **project (Project) :** Référence au projet associé à cette facture.
- **totalcost (float) :** Coût total du projet, probablement la somme de tous les éléments et services.
- **details (String) :** Description détaillée des coûts et des éléments inclus dans la facture.

Méthodes :

- **generateDetail()** :Méthode pour générer et remplir le champ détails avec un résumé des coûts et des informations sur les éléments du projet.
- **calculateTotalCost()** : Méthode pour calculer le coût total du projet. Cette méthode peut mettre à jour l'attribut totalcost avec la dernière valeur calculée.

Classe Dimensions :

La classe **Dimensions** est conçue pour stocker les informations sur les dimensions de la cuisine ou des éléments la constituant dans l'application MyKitchen.

Attributs :

- **width (float) :** La largeur de l'objet ou de l'élément, probablement mesurée en cm.
- **height (float) :** La hauteur de l'objet ou de l'élément mesurée en cm.
- **depth (float) :** La profondeur de l'objet ou de l'élément mesurée en cm.

Méthode :

- **updateDimensions(width: float, height: float, depth: float)** : Met à jour les dimensions de l'objet ou de l'espace avec les nouvelles valeurs fournies.

Classe Kitchen :

La classe **Kitchen** est utilisée pour représenter la configuration d'une cuisine au sein de l'application MyKitchen.

Attributs :

- **dimensions (Dimensions)** : Un objet de la classe Dimensions qui stocke les dimensions physiques de la cuisine, incluant largeur, hauteur et profondeur.
- **elements (List<Element>)** : Une liste d'objets de la classe Element, chacun représentant une composante (ameublement) de la cuisine.

Méthodes :

- **addElement(elementId: Element) : Boolean :** Ajoute un nouvel élément à la liste des éléments de la cuisine. Retourne true si l'ajout est réussi.
- **removeElement(elementId: Element)** : **Boolean :** Supprime un élément existant de la liste des éléments de la cuisine. Retourne true si la suppression est réussie.

- **updateElement(elementId: Element, newElement: Element) : Boolean :** Remplace un élément existant par un nouvel élément dans la liste. Retourne true si la mise à jour est réussie.
- **calculateCost(project: Project, project: Project): float** : Calcule et retourne le coût total d'une cuisine donné.

Classe Element :

La classe **Element** représente un élément individuel ou un composant qui peut être ajouté à une cuisine dans le cadre d'une d'une conception de cuisine.

Attributs :

- **elementID (Integer)** : Identifiant unique pour chaque élément.
- **name (String)** : Le nom de l'élément, qui pourrait être le nom du produit ou une description.
- **type (String)** : Catégorie ou type de l'élément (par exemple, appareil électroménager, meuble, etc.).
- **material (String)** : Matériau de fabrication de l'élément.
- **color (String)** : Couleur de l'élément.
- **price (float)** : Prix de l'élément.
- **size (Dimensions)** : Dimensions de l'élément, représentées par une instance de la classe Dimensions.

Méthode :

- **updateDetails(name: String, type: String, material: String, color: String, price: Real, dimensions: Dimension): Boolean** : Met à jour les détails de l'élément avec les nouveaux paramètres fournis et retourne true si la mise à jour est réussie.

Classe ElementLibraryManager :

La classe **ElementLibraryManager** est conçue pour gérer une collection d'éléments de cuisine, tels que des meubles, des appareils électroménagers, ou d'autres composants utilisés dans la conception de cuisines.

Attribut :

- **elements (List<Element>)** : Une liste qui stocke les éléments disponibles dans la bibliothèque.

Méthodes :

- **searchElements(name: String)** : Recherche dans la liste des éléments ceux qui correspondent au nom donné.
- **addElement(element: List<Element>):** Boolean : Ajoute un ou plusieurs nouveaux éléments à la bibliothèque et retourne true si l'ajout est réussi.
- **removeElement(elementId: Integer**): Boolean : Supprime l'élément correspondant à l'identifiant donné de la bibliothèque et retourne true si la suppression est réussie.
- **updateElement(elementId: Integer, updatedElement: Element**): Boolean : Met à jour les détails d'un élément existant avec un nouvel élément fourni et retourne true si la mise à jour est réussie.

Classe KitchenView :

La classe **KitchenView** agit en tant que contrôleur de l'interface graphique dans une application. Elle orchestre les interactions entre l'utilisateur et l'application.

Méthodes :

- **onLoginButton() :** Action à exécuter lors de l'appui sur le bouton de connexion.
- **onLogoutButton() :** Action à exécuter lors de l'appui sur le bouton de déconnexion.
- **onRegisterClicked() :** Action à exécuter lorsque l'utilisateur clique sur l'option d'enregistrement.
- **onSaveButtonClicked() :** Action à exécuter lorsque l'utilisateur clique sur le bouton de sauvegarde.
- **onDeleteButtonClicked() :** Action à exécuter lorsque l'utilisateur clique sur le bouton de suppression.
- **onSelectButtonClicked() :** Action à exécuter lorsque l'utilisateur clique sur le bouton de sélection.
- **onAddButtonClicked() :** Action à exécuter lorsque l'utilisateur clique sur le bouton d'ajout d'un nouvel élément.

- **onRemoveButtonClicked() :** Action à exécuter lorsque l'utilisateur clique sur le bouton de suppression d'un élément.
- **onUpdateButtonClicked() :** Action à exécuter lorsque l'utilisateur clique sur le bouton de mise à jour d'un élément.
- **onGenerateButtonClicked() :** Action à exécuter lorsque l'utilisateur clique sur le bouton de génération (probablement d'une facture ou d'un devis).
- **onMainMenuButtonClicked() :** Action à exécuter lorsque l'utilisateur retourne au menu principal.
- **onInvoiceButtonClicked() :** Action à exécuter lorsque l'utilisateur souhaite voir la facture d'un projet.
- **showErrorMessage(errormessage: String):** Pour afficher des messages d'erreur à l'utilisateur, par exemple lorsqu'une action échoue ou qu'une entrée n'est pas valide.
- **showSuccessMessage(successmessage: String):** Pour confirmer qu'une action s'est déroulée avec succès.

Classe KitchenDesignerApp :

La classe KitchenDesignerApp est conçue pour être le moteur opérationnel au cœur d'une application de design d'intérieur spécialisée dans les cuisines. Elle orchestre l'ensemble des fonctionnalités clés, offrant une expérience utilisateur fluide et intuitive.

Méthodes :

- **login(username: String, password: String): User** : Authentifie un utilisateur et retourne l'objet User correspondant.
- **logout()** : Déconnecte l'utilisateur actuellement connecté.
- **createProject(user: User): User** : Crée un nouveau projet pour l'utilisateur et retourne l'objet User mis à jour.
- **deleteProject(projectId: Integer): Boolean** : Supprime le projet correspondant à l'ID donné et retourne true si la suppression est réussie.
- **selectProject(projectId: Project): Project** : Sélectionne un projet et retourne l'objet (Project) correspondant.

- **addElement(project: Project, element: Element): Boolean** : Ajoute un élément à un projet donné et retourne true si l'ajout est réussi.
- **removeElement(project: Project, elementId: Element**): Boolean : Supprime un élément d'un projet et retourne true si la suppression est réussie.
- **updateElement(project: Project, elementId: Integer, newElement: Element): Boolean** : Met à jour les détails d'un élément dans un projet et retourne true si la mise à jour est réussie.
- **calculateCost(project: Project, project: Project): float** : Calcule et retourne le coût total d'un projet donné.
- **generateInvoice(project: Project): Invoice** : Génère une facture pour un projet donné et retourne l'objet Invoice.

Classe Administrator :

La classe **Administrator** est conçue comme un pivot de gestion au sein de l'application, armée de privilèges élargis pour superviser des aspects critiques du système.

Attributs :

- **adminId (Integer) :** Un identifiant unique qui distingue chaque administrateur dans le système.
- **username (String) :** Le nom d'utilisateur utilisé par l'administrateur pour se connecter à l'application.

Méthodes :

- **manageUsers(users: List<User>)** : Permet à l'administrateur d'effectuer des opérations sur la liste des utilisateurs, telles que la création, la mise à jour ou la suppression de comptes utilisateurs.
- **updateElementLibrary(element: Element): Boolean :** Offre la possibilité d'ajouter ou de modifier des éléments dans la bibliothèque centrale d'éléments de cuisine, retournant true si l'action est réussie.

- **manageProjects(projects: List<Project>): Boolean :** Confère la capacité de gérer la liste des projets de conception de cuisine, y compris leur création, modification et suppression.
- **modifyElementList(elementId: Integer, updatedElement: Element):** Boolean : Fournit un moyen de mettre à jour spécifiquement les détails d'un élément existant dans la bibliothèque, avec confirmation de succès via un booléen.

Associations:

1) **User - Project** :
- Un User peut avoir zéro ou plusieurs Project(s).
- Chaque Project est associé à exactement un User.
- Cardinalité : 1 à 0..*

2) **Project - Kitchen** :
- Chaque Project contient exactement une Kitchen.
- Chaque Kitchen est associée à exactement un Project.
- Cardinalité : 1 à 1.

3) **Kitchen - Element** :
- Une Kitchen peut avoir zéro ou plusieurs Element(s).
- Chaque Element est associé à exactement une Kitchen.
- Cardinalité : 1 à 0..*.

4) **Project - Invoice** :
- Chaque Project génère exactement une Invoice.
- Chaque Invoice est associée à exactement un Project.
- Cardinalité : 1 à 1.

5) **Kitchen - Dimensions** :
- Chaque Kitchen a exactement un Dimensions.
- Chaque Dimensions est associé à exactement une Kitchen.

- Cardinalité : 1 à 1.

6) **Element - Dimensions** :
- Chaque Element a exactement un Dimensions.
- Chaque Dimensions est associé à exactement un Élément.
- Cardinalité : 1 à 1.

7) **ElementLibraryManager - Element** :
- Un ElementLibraryManager peut gérer zéro ou plusieurs Element(s).
- Chaque Element peut être géré par un ElementLibraryManager.
- Cardinalité : 1 à 0..*.

8) **Administrator-User :**
- Chaque Administrator peut gérer zéro ou plusieurs User.
- Chaque User est placé sous la supervision d'un Administrator.
- Cardinalité pour 0...* à 1.

Remarque : Dans le cadre du diagramme de classes fourni, il est important de noter que l'Administrator hérite des droits de l'User, ce qui lui permet de gérer non seulement les utilisateurs eux-mêmes mais aussi leurs projets. Cette capacité d'héritage implique que l'administrateur peut exercer toutes les actions qu'un utilisateur standard peut effectuer, en plus des responsabilités administratives supplémentaires telles que la gestion de la bibliothèque d'éléments et des projets globaux.

Après avoir établi le diagramme de classe qui décrit la structure interne du système, il est pertinent d'élaborer des scénarios et des diagrammes de séquence pour illustrer concrètement comment ces classes interagissent dans différentes situations d'utilisation. Les scénarios et les diagrammes de séquence permettent d'approfondir la compréhension des fonctionnalités du système en montrant comment les objets interagissent entre eux pour réaliser des actions spécifiques.

En effet, le diagramme de classe fournit une vue statique des classes, de leurs attributs et de leurs méthodes, tandis que les scénarios et les diagrammes de séquence offrent une vue dynamique en décrivant le flux d'interactions entre les objets pendant l'exécution des fonctionnalités du système.

Ainsi, la transition entre le diagramme de classe et les scénarios, diagrammes de séquence permet d'aller au-delà de la simple représentation structurelle du système pour explorer son fonctionnement concret et les interactions entre ses différentes parties dans des situations réalistes d'utilisation.

IV. The possibles uses scenarios :

A) Ce diagramme de séquence démontre la séquence d'interactions de l'utilisateur avec les composantes de l'application MyKitchen, spécifiquement pendant le processus de conception et de sauvegarde d'une cuisine virtuelle. Les étapes suivantes décrivent ce processus en détail :

1) Authentification de l'utilisateur :

- L'utilisateur commence par s'authentifier dans l'application KitchenDesignerApp en fournissant son nom d'utilisateur et son mot de passe (login(Tom, abc123)).
- Le système valide les informations et retourne une confirmation de la connexion réussie (OK).

2) Création d'un projet :

- Après la connexion, l'utilisateur crée un nouveau projet en spécifiant le nom du projet et son propre identifiant (createProject(1, ma nouvelle cuisine, luxeuse)).
- Le système confirme la création du projet (OK).

3) **Saisie des dimensions de la cuisine :**

- L'utilisateur est invité à entrer les dimensions de sa cuisine (Entre les dimensions de votre cuisine).
- Ces informations sont transmises au composant KitchenView qui permet de visualiser la cuisine.

3) Recherche d'éléments à ajouter :

- L'utilisateur recherche des éléments à ajouter à son projet de cuisine (searchElements(four)), comme un four dans cet exemple.
- La recherche est en cours et le système retourne les éléments correspondants.

4) **Sélection et ajout d'éléments :**

- Une fois l'élément trouvé, l'utilisateur sélectionne l'élément désiré en cliquant sur le bouton de sélection (onSelectButtonClicked()).
- Le système confirme la sélection (OK).
- L'utilisateur ajoute ensuite l'élément sélectionné à la cuisine (addElement(1, four à pizza, Four, Pierre, Gris, 2000)).

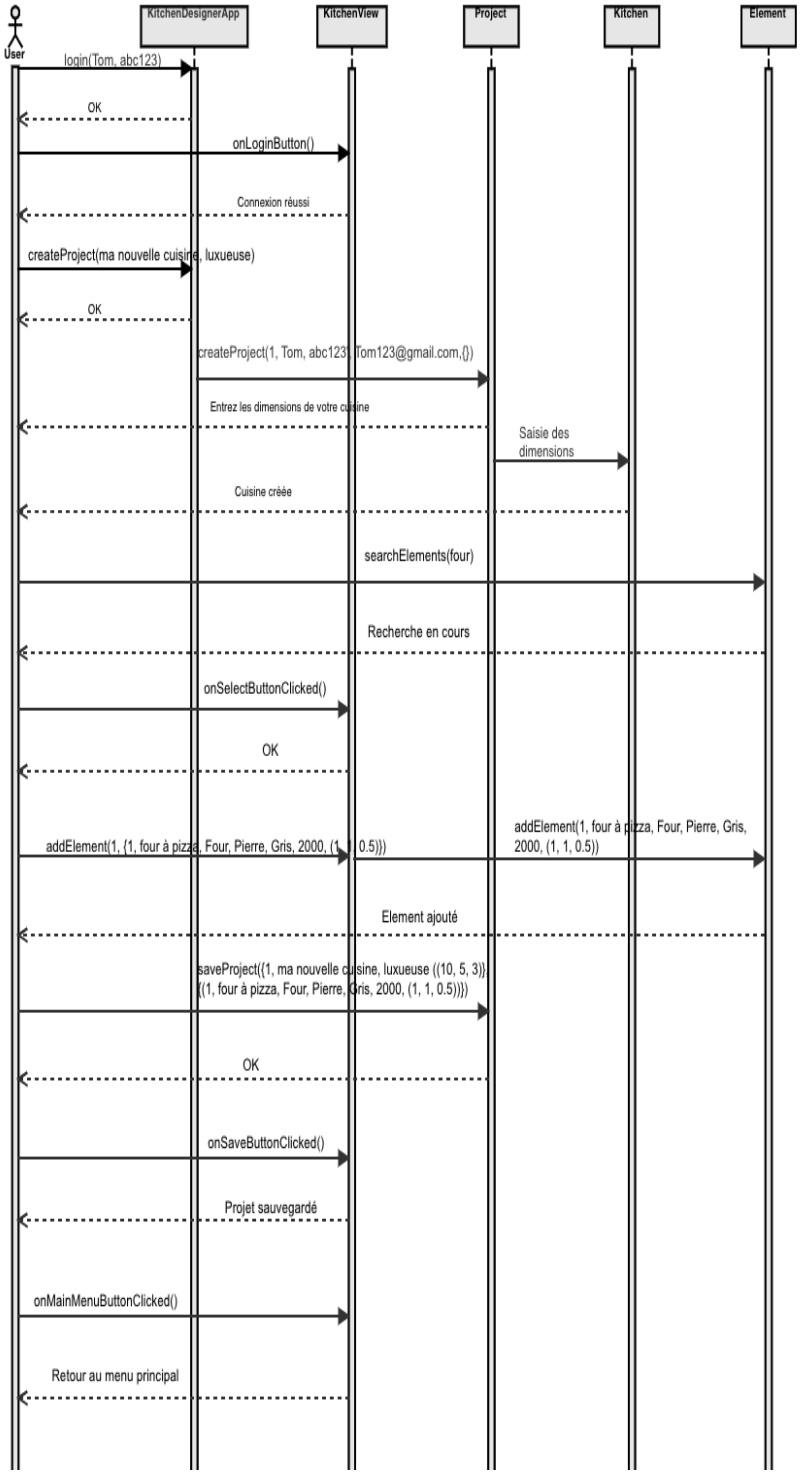
5) **Sauvegarde du projet :**

- Après avoir ajouté les éléments désirés, l'utilisateur sauvegarde le projet (saveProject(1, ma nouvelle cuisine, luxeuse, {(1, four à pizza, Four, Pierre, Gris, 2000)})).
- Le système confirme que le projet a été sauvegardé avec succès (Projet sauvegardé).

6) **Retour au menu principal :**

- Enfin, l'utilisateur décide de retourner au menu principal de l'application en cliquant sur le bouton du menu principal (onMainMenuButtonClicked()).

Ce scénario décrit les interactions typiques d'un utilisateur avec l'application MyKitchen pour démarrer un projet de conception de cuisine, de l'authentification jusqu'à la sauvegarde des choix effectués.



A) Le diagramme présenté retrace le parcours de l'utilisateur qui, ayant débuté la création d'un projet dans MyKitchen, décide de le mettre en pause pour le reprendre ultérieurement. Voici une explication détaillée des différentes étapes de cette séquence :

1) **Authentification :**

- L'utilisateur se connecte à l'application KitchenDesignerApp avec son identifiant et son mot de passe (par exemple login(Tom, abc123)).
- La réponse positive (OK) confirme la réussite de la connexion.

2) **Création d'un projet :**

- L'utilisateur crée un nouveau projet en spécifiant le nom et d'autres détails (par exemple createProject(ma nouvelle cuisine, luxueuse)).
- Le système confirme la création du projet (OK).

3) **Saisie des dimensions de la cuisine :**

- Le système invite l'utilisateur à saisir les dimensions de sa cuisine.
- Ces dimensions sont probablement enregistrées ou utilisées pour configurer l'espace de travail pour la conception.

4) **Reprise du projet :**

- L'utilisateur choisit de reprendre un projet existant (selectProject(1)), indiquant son intention de continuer un travail précédemment commencé.
- Le système confirme l'ouverture du projet (message Ouverture du projet...).

5) **Ajout d'éléments à la cuisine :**

- L'utilisateur procède à la recherche d'éléments à ajouter à son projet (searchElements(four)), comme un four dans cet exemple.
- Après la sélection (onSelectButtonClicked()), il reçoit une confirmation (OK).

- L'utilisateur ajoute ensuite l'élément à son projet (addElement(1, four à pizza, Four, Pierre, Gris, 2000)).
- Le système confirme que l'élément a été ajouté.

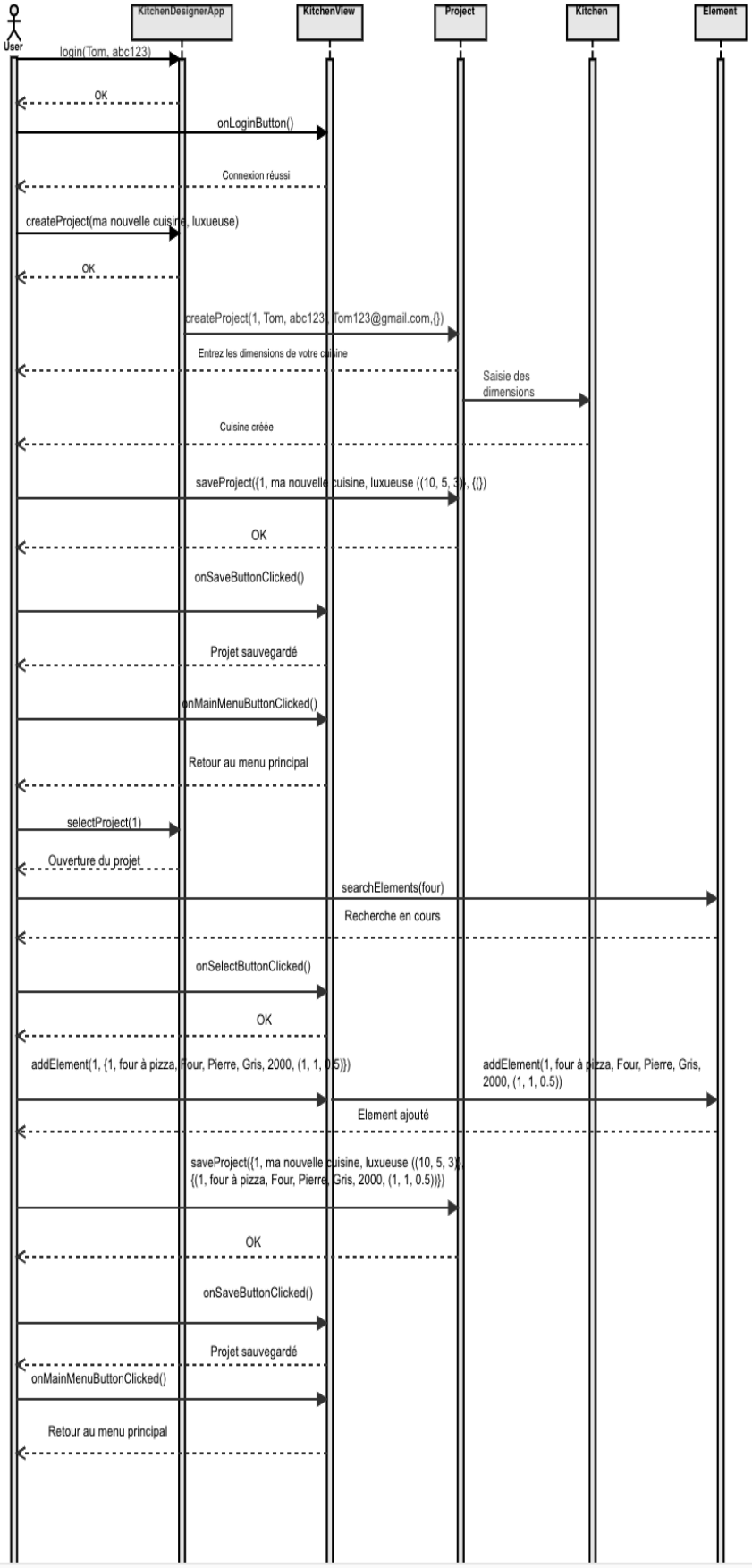
6) **Sauvegarde des modifications :**

- L'utilisateur sauvegarde les changements apportés au projet (saveProject(1, ma nouvelle cuisine luxueuse, {...})).
- Le système confirme la sauvegarde du projet (OK).

7) **Retour au menu principal :**

- Après la sauvegarde, l'utilisateur retourne au menu principal (onMainMenuButtonClicked()), complétant ainsi la session de travail actuelle.

Ce scénario décrit le processus d'interaction de l'utilisateur avec l'application MyKitchen, en mettant l'accent sur la capacité de reprendre un projet de conception de cuisine en cours. Il illustre l'importance d'une expérience utilisateur fluide, permettant de sauvegarder et de reprendre des projets sans perte de données ou d'interruption dans le flux de travail.


User
KitchenDesignerApp
KitchenView
Project
Kitchen
Element
login(Tom, abc123)
OK
onLoginButton()
Connexion réussi
createProject(ma nouvelle cuisine, luxueuse)
OK
createProject(1, Tom, abc123, Tom123@gmail.com,{})
Entrez les dimensions de votre cuisine
Saisie des dimensions
Cuisine créée
saveProject({1, ma nouvelle cuisine, luxueuse ((10, 5, 3), {()}
OK
onSaveButtonClicked()
Projet sauvegardé
onMainMenuButtonClicked()
Retour au menu principal
selectProject(1)
Ouverture du projet
searchElements(four)
Recherche en cours
onSelectButtonClicked()
OK
addElement(1, {1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5)})
addElement(1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5))
Element ajouté
saveProject({1, ma nouvelle cuisine, luxueuse ((10, 5, 3), {{1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5)}})
OK
onSaveButtonClicked()
Projet sauvegardé
onMainMenuButtonClicked()
Retour au menu principal

B) Ce diagramme de séquence met en scène un utilisateur qui, après avoir ajouté un élément à sa cuisine dans l'application MyKitchen, choisit de le retirer, puis, insatisfait du projet global, décide de repartir de zéro en supprimant le projet en cours. Les étapes ci-dessous expliquent ce processus :

1) **Connexion de l'utilisateur :**

- L'utilisateur se connecte à l'application KitchenDesignerApp en utilisant ses identifiants (login(Tom, abc123)).
- La confirmation de la connexion réussie est renvoyée (OK).

2) **Création d'un projet de cuisine :**

- L'utilisateur crée un nouveau projet de cuisine (createProject(ma nouvelle cuisine, luxueuse)).
- Le système renvoie un message de confirmation (OK).

3) **Ouverture d'un projet existant :**

- L'utilisateur choisit d'ouvrir un projet existant (selectProject(1)).
- Le système confirme l'ouverture du projet (Ouverture du projet...).

4) **Ajout d'un élément à la cuisine :**

- L'utilisateur ajoute un nouvel élément, par exemple un four à pizza (addElement(1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5))).
- Le système confirme que l'élément a été ajouté (Element ajouté).

5) **Suppression d'un élément de la cuisine :**

- L'utilisateur décide qu'il n'aime pas l'élément et le supprime (removeElement(1, 1)).

- Le système confirme la suppression de l'élément (OK).

6) **Suppression du projet de cuisine :**

- Après réflexion, l'utilisateur décide de supprimer le projet entier car il n'est pas satisfait de la conception (deleteProject(1)).
- Le système exécute la suppression et renvoie l'utilisateur au menu principal.

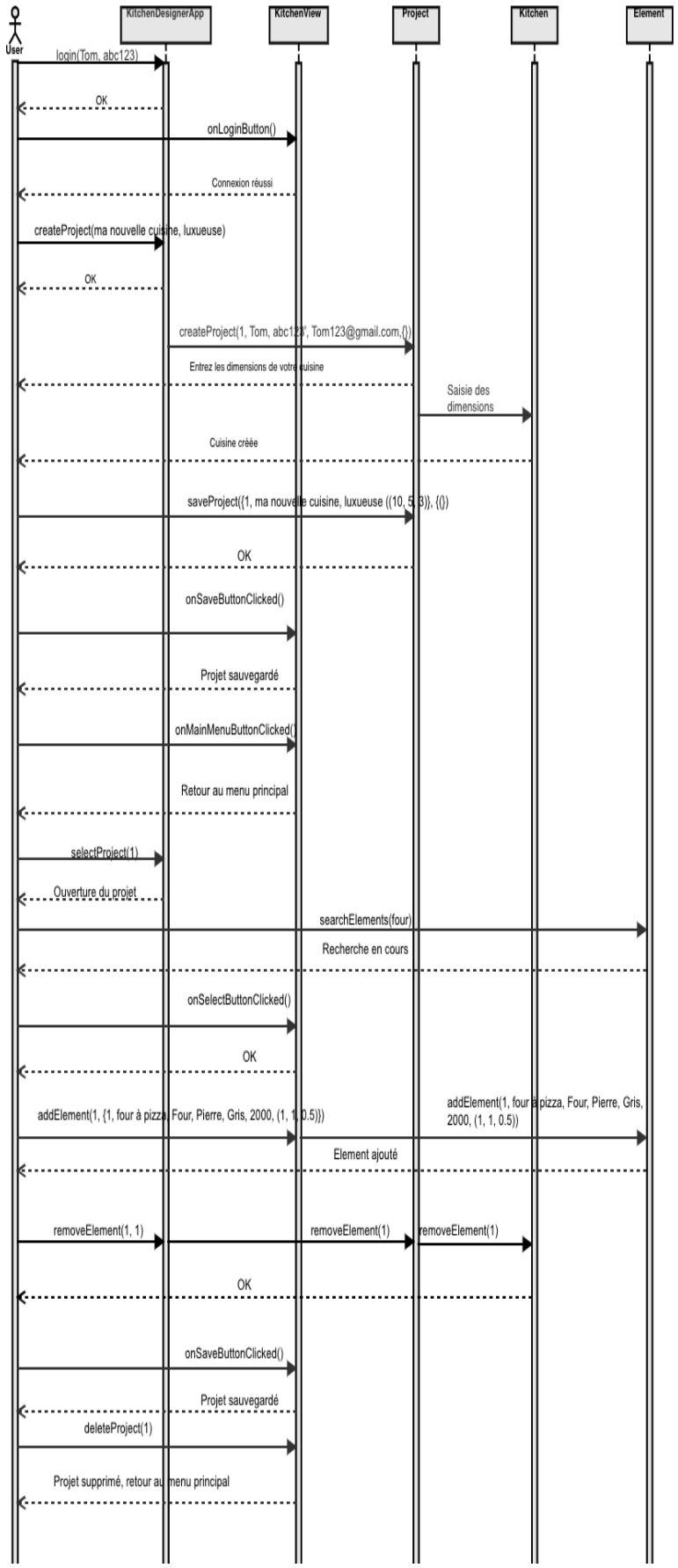
7) **Sauvegarde et fermeture :**

Avant de quitter, l'utilisateur sauvegarde l'état actuel du projet (onSaveButtonClicked()), bien que dans ce scénario cela puisse ne pas être nécessaire puisque le projet est supprimé.
Le système confirme la sauvegarde du projet (Projet sauvegardé).

8) **Retour au menu principal :**

- L'utilisateur retourne au menu principal de l'application (onMainMenuButtonClicked()), potentiellement pour commencer un nouveau projet.

Ce scénario illustre le flux d'actions que l'utilisateur peut effectuer lorsqu'il travaille sur un projet dans l'application MyKitchen, notamment l'ajout et la suppression d'éléments, ainsi que la suppression d'un projet entier lorsqu'il souhaite recommencer la conception de sa cuisine.


User
KitchenDesignerApp
KitchenView
Project
Kitchen
Element
login(Tom, abc123)
OK
onLoginButton()
Connexion réussi
createProject(ma nouvelle cuisine, luxueuse)
OK
createProject(1, Tom, abc123", Tom123@gmail.com,{})
Entrez les dimensions de votre cuisine
Saisie des dimensions
Cuisine créée
saveProject({1, ma nouvelle cuisine, luxueuse ((10, 5, 3)), {()}
OK
onSaveButtonClicked()
Projet sauvegardé
onMainMenuButtonClicked()
Retour au menu principal
selectProject(1)
Ouverture du projet
searchElements(four)
Recherche en cours
onSelectButtonClicked()
OK
addElement(1, {1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5)})
addElement(1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5))
Element ajouté
removeElement(1, 1)
removeElement(1)
removeElement(1)
OK
onSaveButtonClicked()
Projet sauvegardé
deleteProject(1)
Projet supprimé, retour au menu principal

C) À travers ce diagramme de séquence, nous observons les étapes franchies par l'utilisateur qui, satisfait de la conception de sa cuisine virtuelle dans MyKitchen et du coût associé, procède à la sauvegarde de son projet et à la génération de la facture correspondante. Voici un déroulé détaillé des actions effectuées :

1) **Connexion de l'utilisateur :**

- L'utilisateur se connecte à l'application KitchenDesignerApp avec son identifiant et mot de passe (login(Tom, abc123)).
- Le système renvoie un message "OK" pour confirmer la connexion.

2) **Création et configuration du projet de cuisine :**

- L'utilisateur crée un nouveau projet de cuisine (createProject(ma nouvelle cuisine, luxueuse)).
- Après la création, le système demande à l'utilisateur d'entrer les dimensions de la cuisine.
- L'utilisateur entre les dimensions et commence la création de la cuisine, avec l'ajout d'éléments comme un four à pizza (addElement(1, four à pizza, Four, Pierre, Gris, 2000, (1, 0.5))).
- Le système confirme que l'élément a été ajouté.

3) **Évaluation du coût :**

- L'utilisateur demande le calcul du coût du projet (calculateCost(1)).
- Le système calcule le coût et renvoie l'information que "Le prix total est de 2000€", ce qui est jugé raisonnable par l'utilisateur.

4) **Sauvegarde du projet :**

- L'utilisateur décide de sauvegarder le projet (saveProject(1, ma nouvelle cuisine, luxueuse, {...})).

- Le système confirme la sauvegarde avec un "OK".

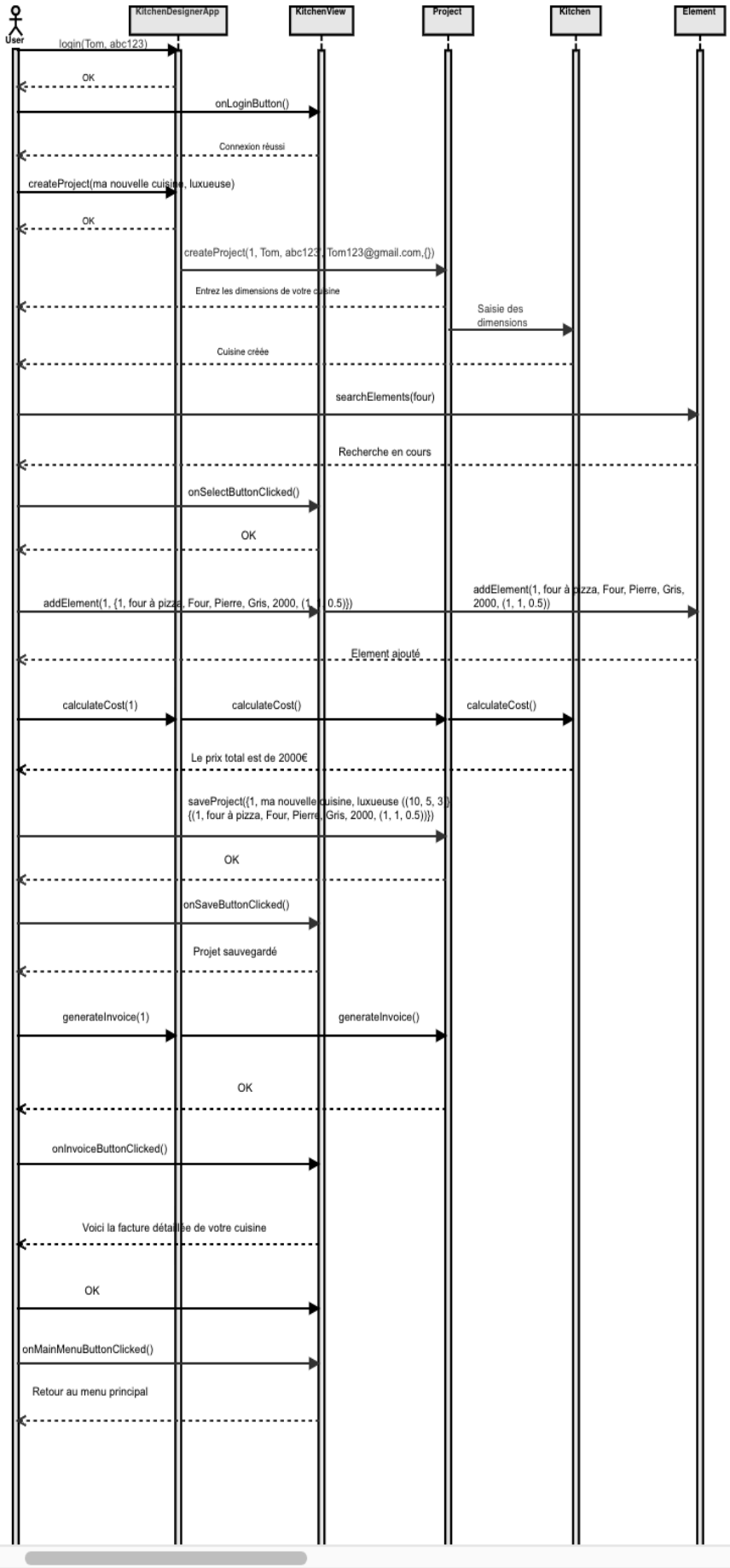
5) **Génération de la facture :**

- Ensuite, l'utilisateur génère la facture pour le projet de cuisine (generateInvoice(1)).
- Le système crée la facture et renvoie un message "Voici la facture détaillée de votre cuisine".

6) **Clôture et retour au menu principal :**

- L'utilisateur achève le processus en cliquant sur le bouton pour revenir au menu principal (onMainMenuButtonClicked()).
- Le système renvoie l'utilisateur au menu principal, terminant ainsi la session de conception de la cuisine.

Ce scénario illustre le processus complet que l'utilisateur suit dans l'application KitchenDesignerApp pour concevoir une cuisine, de la connexion à la génération de la facture finale, en passant par l'évaluation des coûts et la sauvegarde du projet.


User
KitchenDesignerApp
KitchenView
Project
Kitchen
Element
login(Tom, abc123)
OK
onLoginButton()
Connexion réussi
createProject(ma nouvelle cuisine, luxueuse)
OK
createProject(1, Tom, abc123, Tom123@gmail.com,{})
Entrez les dimensions de votre cuisine
Saisie des dimensions
Cuisine créée
searchElements(four)
Recherche en cours
onSelectButtonClicked()
OK
addElement(1, {1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5)})
addElement(1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5))
Element ajouté
calculateCost(1)
calculateCost()
calculateCost()
Le prix total est de 2000€
saveProject({1, ma nouvelle cuisine, luxueuse ({10, 5, 3}, {{1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5)}})
OK
onSaveButtonClicked()
Projet sauvegardé
generateInvoice(1)
generateInvoice()
OK
onInvoiceButtonClicked()
Voici la facture détaillée de votre cuisine
OK
onMainMenuButtonClicked()
Retour au menu principal

***E)* Ce diagramme de séquence démontre la séquence d'interactions de l'utilisateur avec les composantes de l'application MyKitchen en basculant entre deux sessions, une nouvelle et une déjà existante.**

1) ***Authentification***

- L'utilisateur fournit son nom d'utilisateur (Tom) et son mot de passe (abc123) pour s'authentifier dans l'application KitchenDesignerApp.
- Le système vérifie les informations et confirme la connexion réussie (OK).

2) ***Création d'un Projet***

- Après l'authentification, l'utilisateur crée un nouveau projet en spécifiant le nom du projet ("Ma Nouvelle Cuisine") et son identifiant (1).
- Le système confirme la création du projet (OK).

3) ***Saisie des Dimensions***

- L'utilisateur entre les dimensions de sa cuisine pour démarrer la conception.
- Ces informations sont transmises au composant KitchenView pour la visualisation de la cuisine.

4) ***Calcul du Prix***

- Le système calcule le prix total des éléments dans la cuisine, mais il reste à zéro car aucun élément n'a été ajouté.
- L'utilisateur peut visualiser le coût actuel de la cuisine.

5) ***Retour au Menu Principal***

- Enfin, l'utilisateur décide de retourner au menu principal en sélectionnant l'option correspondante.
- Le système le redirige vers le menu principal de l'application.

6) ***Sélection d'un Projet Existant***

- L'utilisateur choisit un projet déjà existant parmi ceux disponibles.

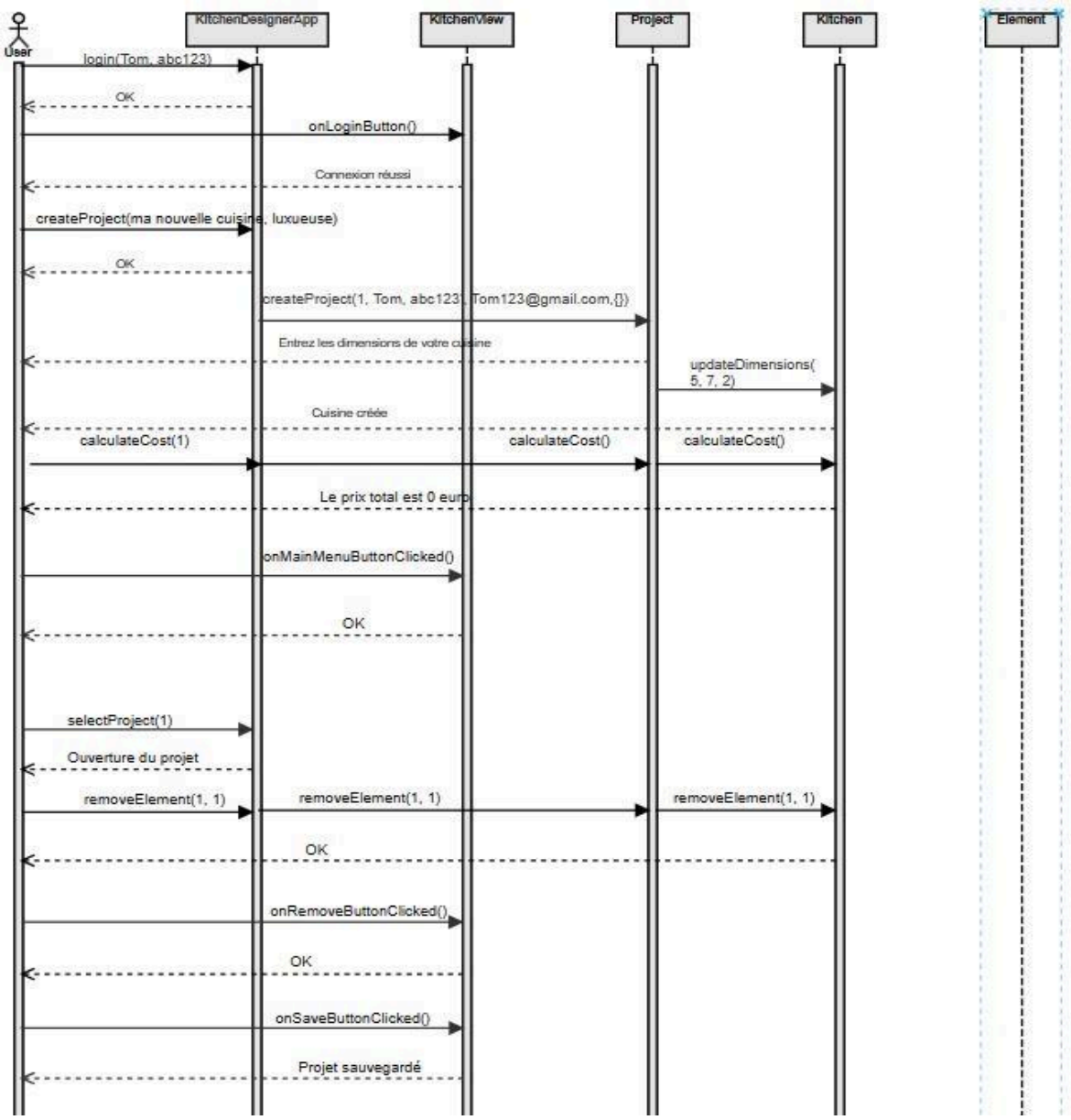
- Le système charge les détails du projet sélectionné pour permettre des modifications ultérieures.

7) Enlèvement d'un Élément

- L'utilisateur supprime un élément spécifique du projet en cours.
- Le système met à jour le projet en retirant l'élément sélectionné.

8) Sauvegarde du Projet

- Après avoir apporté des modifications, l'utilisateur sauvegarde le projet pour enregistrer les changements effectués.
- Le système confirme la sauvegarde réussie du projet, garantissant ainsi la conservation des modifications apportées.



***F)* Ce diagramme de séquence raconte la récupération d'un projet déjà existant la liste disponible, sa modification et puis sa suppression vu que l'utilisateur n'en voulait plus.**

1) ***Authentification***

- L'utilisateur fournit ses informations d'identification (nom d'utilisateur : Tom, mot de passe : abc123) pour accéder à l'application KitchenDesignerApp.
- Le système vérifie les informations fournies et autorise l'accès à l'application (OK).

2) ***Sélection d'un Projet Existant***

- L'utilisateur sélectionne un projet existant parmi ceux disponibles dans son compte.
- Le système charge les détails du projet sélectionné pour permettre des modifications ultérieures.

3) ***Recherche et Sélection d'un Premier Élément***

- L'utilisateur lance une recherche pour trouver le premier élément à intégrer dans sa cuisine, tel qu'un "four".
- Le système effectue la recherche et affiche les résultats correspondants.
- L'utilisateur sélectionne le four parmi les résultats de la recherche.
- Le système confirme la sélection de l'évier et le prépare pour l'ajout à l'espace de cuisine.

4) ***Ajout du Premier Élément***

- L'utilisateur ajoute le four à l'espace de cuisine qu'il est en train de concevoir.
- Le système intègre le four à l'espace de cuisine et met à jour les informations du projet.

5) ***Recherche et Sélection d'un Deuxième Élément***

- L'utilisateur lance une nouvelle recherche pour trouver un deuxième élément à ajouter, comme une "table de cuisson à gaz".
- Le système exécute la recherche et présente les résultats à l'utilisateur.
- L'utilisateur choisit la table de cuisson à gaz parmi les éléments proposés.
- Le système confirme la sélection de la table de cuisson à gaz.

6) ***Ajout du Deuxième Élément***

- L'utilisateur ajoute la table de cuisson à gaz à l'espace de cuisine en cours de conception.
- Le système intègre la table de cuisson à gaz à l'espace de cuisine et met à jour les informations du projet.

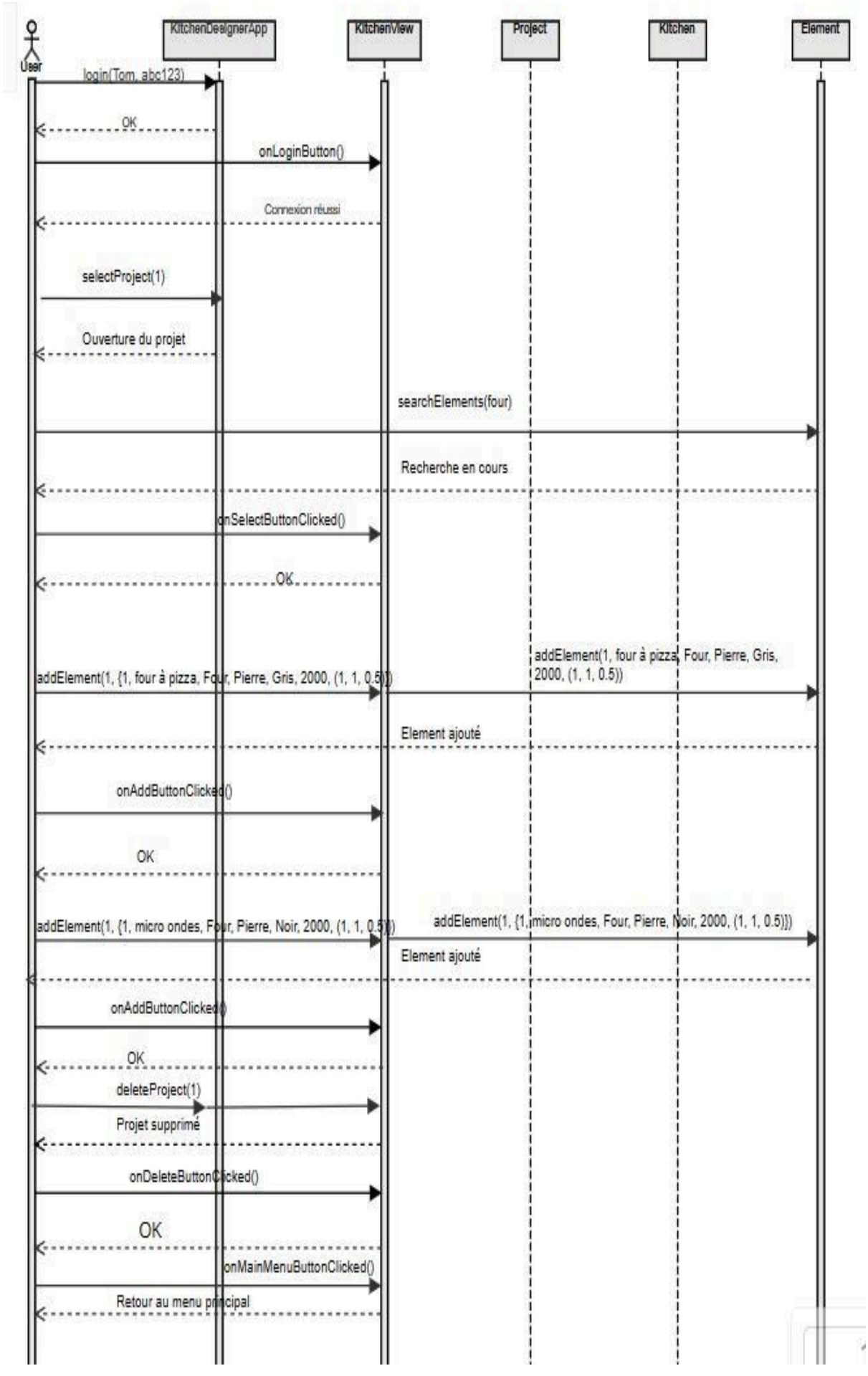
7) ***Suppression du Projet***

- Après avoir terminé ses modifications, l'utilisateur supprime le projet en cours.
- Le système confirme la suppression du projet et nettoie toutes les données associées.

8) ***Retour au Menu Principal***

- Enfin, l'utilisateur décide de retourner au menu principal de l'application en sélectionnant l'option correspondante.
- Le système redirige l'utilisateur vers le menu principal pour explorer d'autres fonctionnalités ou créer de nouveaux projets.

Ce scénario décrit de manière détaillée les étapes impliquées dans la conception d'une cuisine virtuelle avec l'application MyKitchen, mettant en évidence l'interaction fluide entre l'utilisateur et le système.


User
KitchenDesignerApp
KitchenView
Project
Kitchen
Element
login(Tom, abc123)
OK
onLoginButton()
Connexion réussi
selectProject(1)
Ouverture du projet
searchElements(four)
Recherche en cours
onSelectButtonClicked()
OK
addElement(1, {1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5)})
addElement(1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5))
Element ajouté
onAddButtonClicked()
OK
addElement(1, {1, micro ondes, Four, Pierre, Noir, 2000, (1, 1, 0.5)})
addElement(1, {1, micro ondes, Four, Pierre, Noir, 2000, (1, 1, 0.5)})
Element ajouté
onAddButtonClicked()
OK
deleteProject(1)
Projet supprimé
onDeleteButtonClicked()
OK
onMainMenuButtonClicked()
Retour au menu principal

***G)* Ce diagramme de séquence récite un ensemble d'interactions utilisateur - système qui montre un processus de modification d'un projet et la déconnexion vers la fin**

1) ***Authentification***

- L'utilisateur entre ses informations d'identification (nom d'utilisateur : Tom, mot de passe : abc123) pour accéder à l'application KitchenDesignerApp.
- Le système vérifie les informations fournies et permet à l'utilisateur d'accéder à son compte (OK).

2) ***Création d'un Projet***

- Une fois connecté, l'utilisateur crée un nouveau projet en spécifiant son nom ("Projet Cuisine Moderne") et son identifiant unique.
- Le système enregistre le nouveau projet dans la base de données et confirme la création (OK).

3) ***Saisie des Dimensions***

- L'utilisateur saisit les dimensions de la cuisine qu'il souhaite concevoir, fournissant des détails cruciaux pour la planification.
- Les dimensions sont enregistrées dans le projet en cours pour une utilisation ultérieure.

4) ***Recherche et Sélection d'un Élément***

- L'utilisateur lance une recherche pour trouver un élément spécifique à ajouter à sa cuisine, tel qu'une "micro ondes".
- Le système exécute la recherche et affiche les résultats pertinents.
- L'utilisateur choisit l'évier en granite parmi les options disponibles.

5) ***Ajout de l'Élément à l'Espace Créé***

- L'utilisateur ajoute l'évier en granite à l'espace de la cuisine en cours de conception.
- Le système intègre l'évier à la conception de la cuisine et met à jour les informations du projet.

6) ***Sauvegarde du Projet***

- Après avoir effectué des modifications, l'utilisateur sauvegarde le projet pour enregistrer ses progrès.
- Le système confirme la sauvegarde réussie du projet, préservant ainsi les modifications apportées.

7) ***Génération de la Facture***

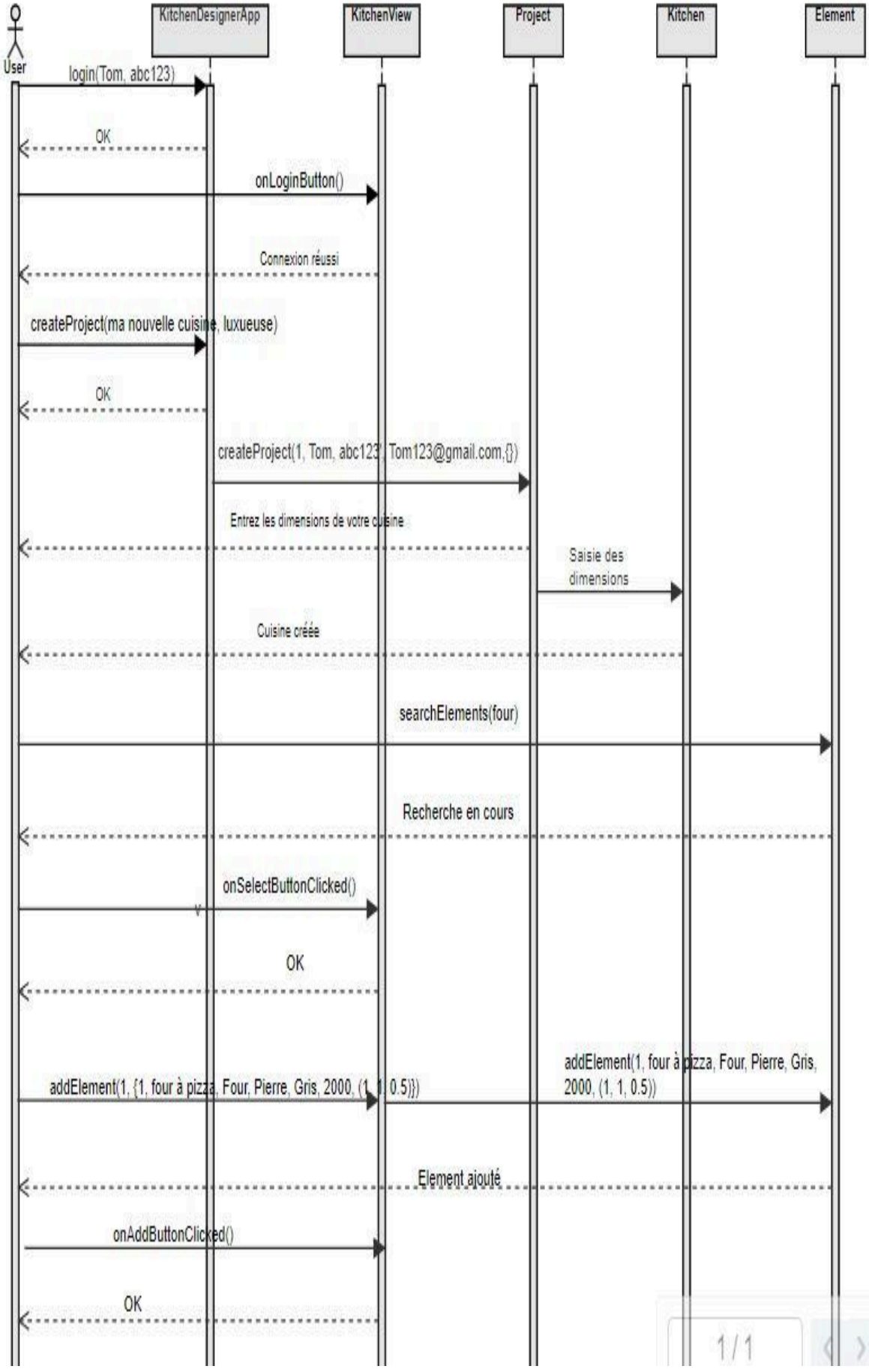
- L'utilisateur demande au système de générer la facture pour le projet de cuisine moderne.
- Le système calcule le coût total du projet et génère une facture détaillée pour l'utilisateur.

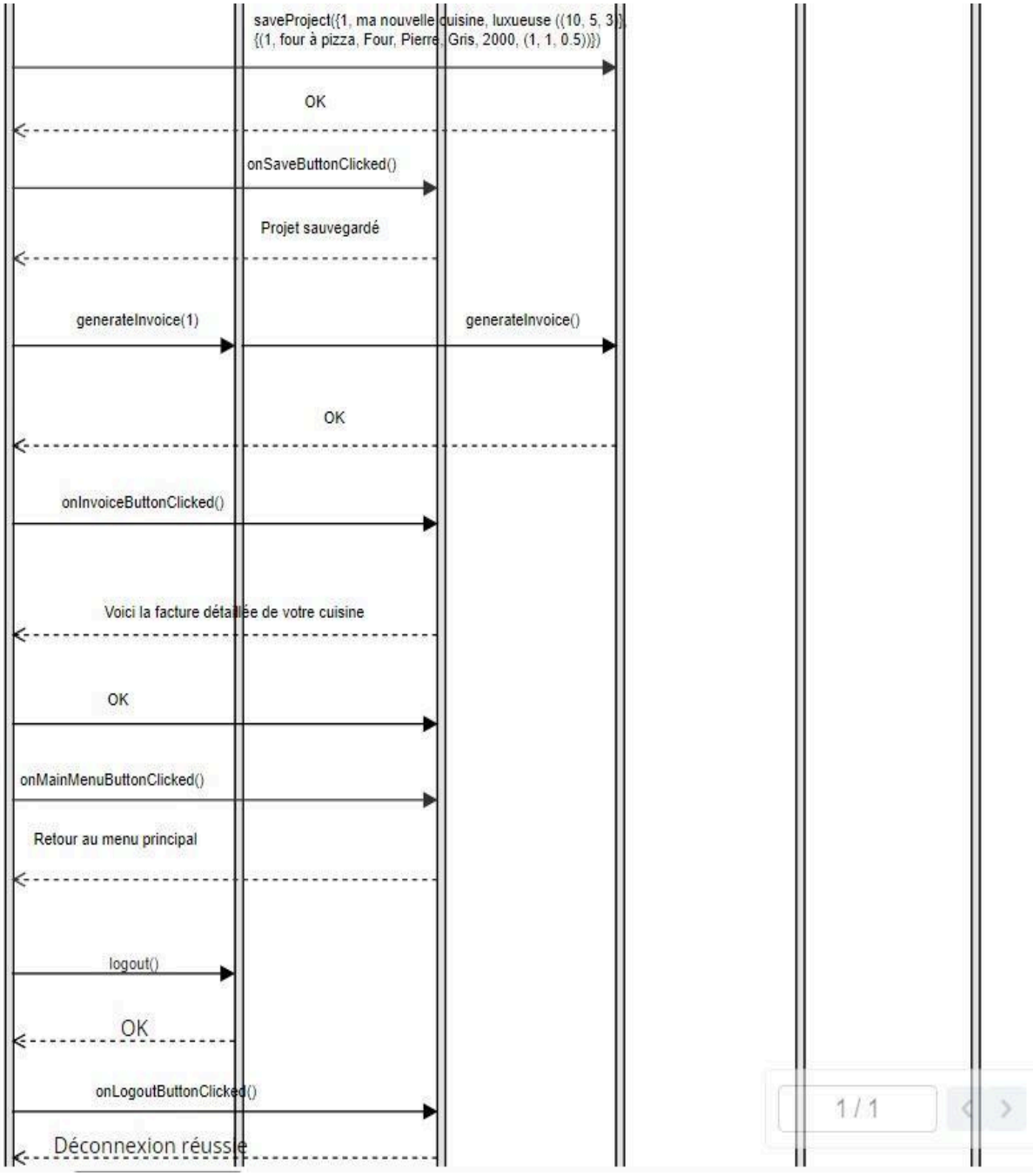
8) ***Retour au Menu Principal***

- Enfin, l'utilisateur décide de retourner au menu principal de l'application pour explorer d'autres fonctionnalités ou créer de nouveaux projets.
- Le système redirige l'utilisateur vers le menu principal de l'application.

9) ***Clic sur le Bouton “Logout” et Déconnexion***

- L'utilisateur clique sur le bouton de déconnexion pour se déconnecter de son compte.
- Le système termine la session de l'utilisateur et le déconnecte en toute sécurité.


User
KitchenDesignerApp
KitchenView
Project
Kitchen
Element
login(Tom, abc123)
OK
onLoginButton()
Connexion réussi
createProject(ma nouvelle cuisine, luxueuse)
OK
createProject(1, Tom, abc123, Tom123@gmail.com,{})
Entrez les dimensions de votre cuisine
Saisie des dimensions
Cuisine créée
searchElements(four)
Recherche en cours
onSelectButtonClicked()
OK
addElement(1, {1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5)})
addElement(1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5))
Element ajouté
onAddButtonClicked()
OK



H) **Ce diagramme de séquence met en scène la création d'un projet, la récupération d'un deuxième sans la sauvegarde du premier et sa modification (Ajout et Suppression d'éléments), puis la suppression totale du projet vu que l'utilisateur n'était pas satisfait et enfin la déconnexion**

1) ***Authentification***

- L'utilisateur entre ses informations d'identification (nom d'utilisateur : Tom, mot de passe : abc123) pour accéder à l'application KitchenDesignerApp.
- Le système vérifie les informations fournies et autorise l'accès à l'interface principale (OK).

2) Création d'un Projet

- L'utilisateur crée un nouveau projet de cuisine en spécifiant son nom ("Projet Cuisine de Rêve") et d'autres détails pertinents.
- Le système enregistre le nouveau projet dans la base de données et confirme la création (OK).

3) Saisie des Dimensions

- L'utilisateur entre les dimensions de la cuisine à concevoir, fournissant des données cruciales pour le processus de conception.
- Les dimensions sont enregistrées dans le projet pour référence future.

4) Retour au Menu Principal

- Après avoir saisi les dimensions, l'utilisateur décide de revenir au menu principal pour explorer d'autres fonctionnalités ou créer de nouveaux projets.
- Le système redirige l'utilisateur vers le menu principal de l'application.

5) Sélection d'un Projet Existant

- L'utilisateur sélectionne un projet déjà existant parmi ceux disponibles dans son compte.
- Le système charge les détails du projet sélectionné pour permettre des modifications ultérieures.

6) Recherche d'un Élément

- L'utilisateur lance une recherche pour trouver un élément spécifique à intégrer dans son projet, comme un "plan de travail en granit".
- Le système exécute la recherche et affiche les résultats correspondants.

7) ***Ajout de l'Élément au Projet Ouvert***

- L'utilisateur sélectionne le plan de travail en granit parmi les options de recherche.
- Le système ajoute l'élément sélectionné au projet de cuisine ouvert et met à jour les informations du projet.

8) ***Suppression d'un Élément Existant***

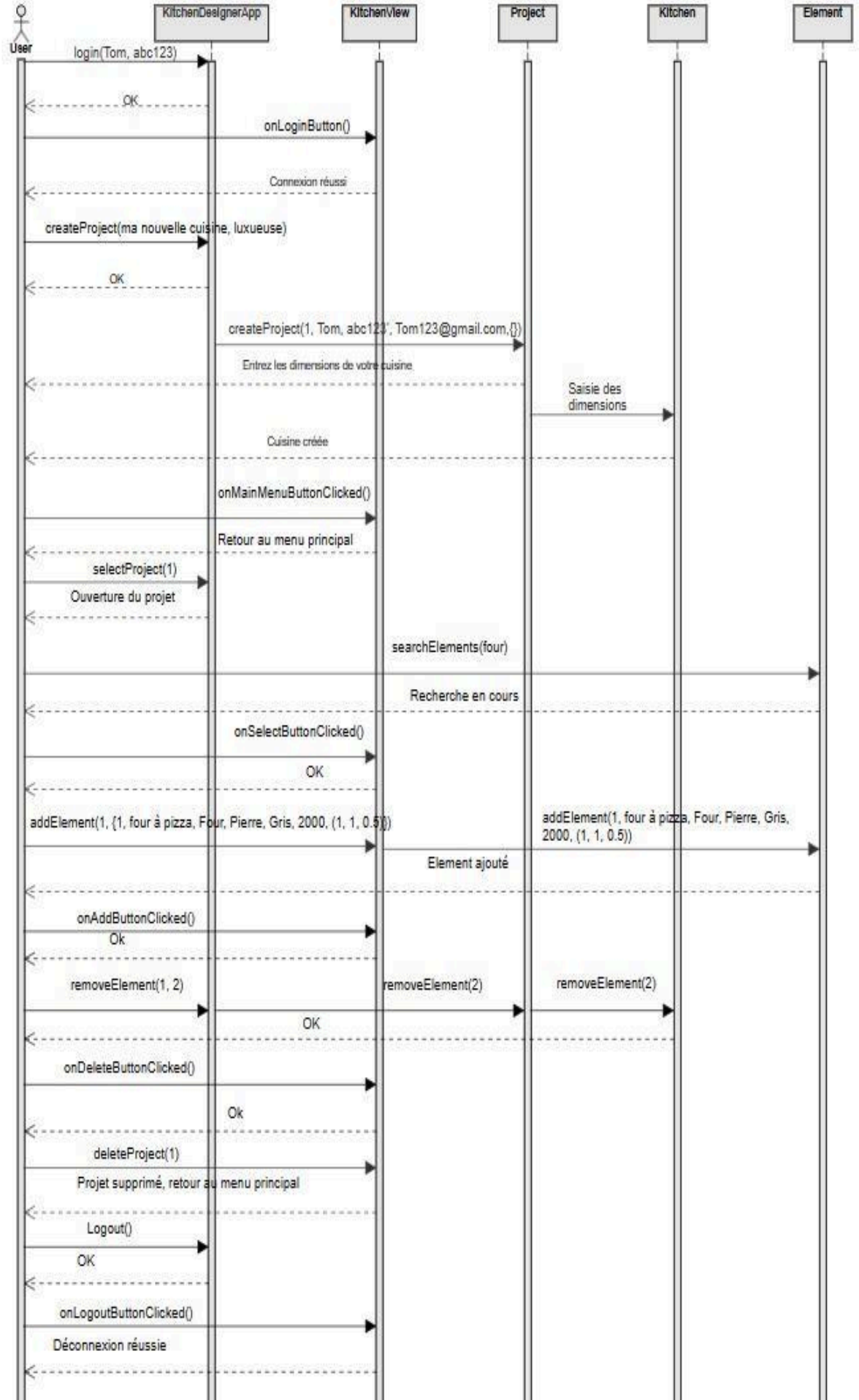
- L'utilisateur choisit de supprimer un élément déjà présent dans le projet en cours.
- Le système supprime l'élément sélectionné du projet et met à jour les détails du projet.

9) ***Suppression du Projet en Cours***

- Après avoir terminé ses modifications, l'utilisateur décide de supprimer le projet actuellement ouvert.
- Le système confirme la suppression du projet et nettoie toutes les données associées.

10) ***Clic sur le Bouton “Logout” et Déconnexion***

- Enfin, l'utilisateur clique sur le bouton de déconnexion pour se déconnecter de son compte.
- Le système termine la session de l'utilisateur et le déconnecte de l'application en toute sécurité.


User
KitchenDesignerApp
KitchenView
Project
Kitchen
Element
login(Tom, abc123)
OK
onLoginButton()
Connexion réussi
createProject(ma nouvelle cuisine, luxueuse)
OK
createProject(1, Tom, abc123', Tom123@gmail.com,{})
Entrez les dimensions de votre cuisine
Saisie des dimensions
Cuisine créée
onMainMenuButtonClicked()
Retour au menu principal
selectProject(1)
Ouverture du projet
searchElements(four)
Recherche en cours
onSelectButtonClicked()
OK
addElement(1, {1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5)})
addElement(1, four à pizza, Four, Pierre, Gris, 2000, (1, 1, 0.5))
Element ajouté
onAddButtonClicked()
Ok
removeElement(1, 2)
removeElement(2)
removeElement(2)
OK
onDeleteButtonClicked()
Ok
deleteProject(1)
Projet supprimé, retour au menu principal
Logout()
OK
onLogoutButtonClicked()
Déconnexion réussie

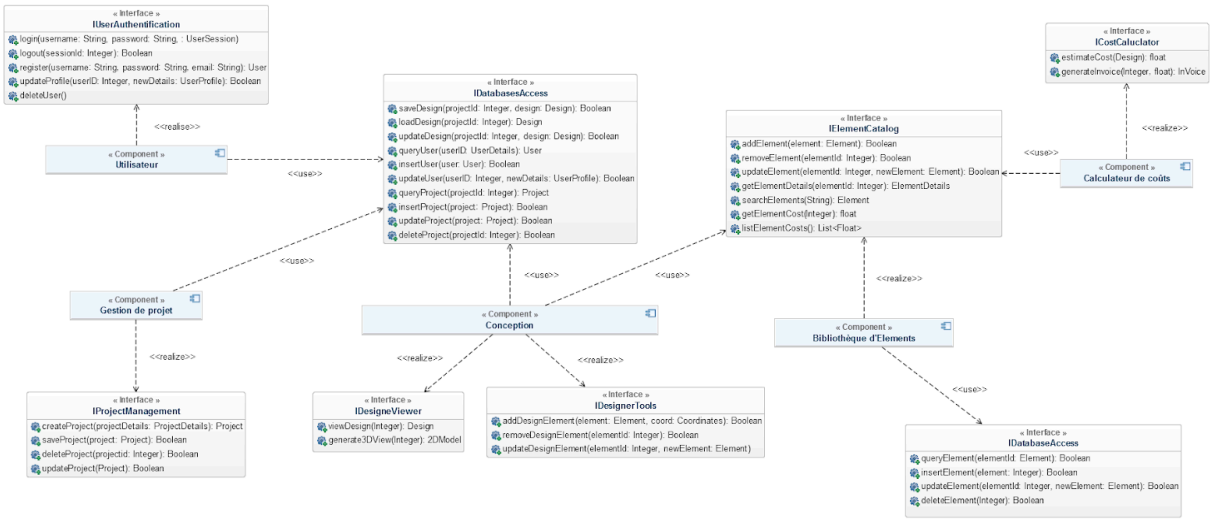
Après avoir établi la structure interne du système à l'aide du diagramme de classe et exploré les interactions entre les objets du système à travers les scénarios et les diagrammes de séquence, il est pertinent de se pencher sur l'architecture logicielle et la distribution des différentes parties du système. C'est là qu'intervient le diagramme de composants.

Le diagramme de composants permet de visualiser les composants logiciels du système et leurs relations, en mettant l'accent sur la façon dont ils sont organisés et interconnectés pour réaliser les fonctionnalités offertes par le système. Il fournit une vue globale de l'architecture logicielle du système, en montrant comment les différentes parties du système sont déployées et comment elles communiquent entre elles.

En relation avec les diagrammes de classe et les scénarios, diagrammes de séquence, le diagramme de composants permet de comprendre comment les classes et les objets du système sont organisés en composants logiciels, et comment ces composants interagissent pour réaliser les fonctionnalités du système dans différents contextes d'utilisation.

Ainsi, la transition vers le diagramme de composants permet d'approfondir la compréhension de l'architecture logicielle du système et de voir comment les différentes parties du système sont organisées et interconnectées pour offrir une expérience utilisateur cohérente et intuitive.

V. Component Diagram (Other Diagram) :



1. Utilisateur

Interfaces requises (<<use>>):

IDataBaseAccess

- **Méthodes :**

  queryUser(userID: UserDetails): User

  - **Paramètres :** userID de type UserDetails pour spécifier l'identifiant de l'utilisateur à récupérer.
  - **Retourne :** Un objet User contenant les détails de l'utilisateur demandé.
  - **Utilité :** Permet de récupérer les informations d'un utilisateur spécifique depuis la base de données pour des opérations comme l'authentification ou la visualisation du profil.

  insertUser(user: User): Boolean

  - **Paramètres :** user de type User contenant les informations de l'utilisateur à ajouter.
  - **Retourne :** Boolean indiquant le succès ou l'échec de l'opération.
  - **Utilité :** Utilisé pour ajouter un nouvel utilisateur dans la base de données, typiquement lors de l'enregistrement d'un nouvel utilisateur.

updateUser(userID: Integer, newDetails: UserProfile): Boolean

- **Paramètres :** userID de type Integer pour l'identifiant de l'utilisateur, et newDetails de type UserProfile contenant les nouvelles informations à mettre à jour.
- **Retourne :** Boolean indiquant le succès ou l'échec de la mise à jour.
- **Utilité :** Permet de mettre à jour les informations d'un utilisateur existant, comme le changement de mot de passe ou la mise à jour de l'adresse e-mail.

deleteUser(Integer): Boolean

- **Paramètres :** Integer représentant l'identifiant de l'utilisateur à supprimer.
- **Retourne :** Boolean indiquant le succès ou l'échec de la suppression.
- **Utilité :** Utilisé pour supprimer un utilisateur de la base de données, généralement à la demande de l'utilisateur ou pour des raisons administratives.
- **Interfaces fournies (<<realize>>):**

IUserAuthentification

- **Méthodes :**

login(username: String, password: String): UserSession

- **Paramètres :** username de type String pour le nom d'utilisateur, et password de type String pour le mot de passe.
- **Retourne :** Un objet UserSession qui représente une session active pour l'utilisateur.
- **Utilité :** Permet à un utilisateur de s'authentifier dans le système en vérifiant ses identifiants. Si l'authentification réussit, une session est initiée.

logout(sessionId: Integer): Boolean

- **Paramètres :** sessionId de type Integer représentant l'identifiant de la session à fermer.
- **Retourne :** Boolean indiquant le succès ou l'échec de la déconnexion.

- **Utilité :** Permet à un utilisateur de se déconnecter du système, terminant ainsi sa session active.

register(username: String, password: String, email: String): User

- **Paramètres :** username de type String, password de type String, et email de type String.
- **Retourne :** Un objet User représentant le nouvel utilisateur enregistré.
- **Utilité :** Permet à un nouvel utilisateur de créer un compte dans le système en fournissant un nom d'utilisateur, un mot de passe et une adresse e-mail.

updateProfile(userID: Integer, newDetails: UserProfile): Boolean

- **Paramètres :** userID de type Integer pour l'identifiant de l'utilisateur, et newDetails de type UserProfile pour les nouvelles informations du profil.
- **Retourne :** Boolean indiquant le succès ou l'échec de la mise à jour du profil.
- **Utilité :** Permet à un utilisateur de mettre à jour son profil, y compris des informations telles que l'adresse e-mail, le numéro de téléphone, etc.

deleteUser()

- **Paramètres :** Aucun paramètre spécifique n'est mentionné, mais on peut supposer qu'il nécessiterait l'identifiant de l'utilisateur ou une session active.
- **Retourne :** Boolean indiquant le succès ou l'échec de la suppression de l'utilisateur.
- **Utilité :** Permet la suppression d'un compte utilisateur, généralement à la demande de l'utilisateur lui-même.

2. Gestion de projet

Interfaces requises (<<use>>):

IDatabaseAccess (spécifique à la gestion de projet)

- **Méthodes :**

queryProject(projectId: Integer): Project

- **Paramètres :** projectId de type Integer pour l'identifiant du projet à récupérer.
- **Retourne :** Un objet Project contenant les détails du projet demandé.
- **Utilité :** Permet de récupérer les informations d'un projet spécifique depuis la base de données pour des opérations comme la visualisation ou la mise à jour du projet.

insertProject(project: Project): Boolean

- **Paramètres :** project de type Project contenant les informations du projet à ajouter.
- **Retourne :** Boolean indiquant le succès ou l'échec de l'opération.
- **Utilité :** Utilisé pour ajouter un nouveau projet dans la base de données, typiquement lors de la création d'un nouveau projet par un utilisateur.

updateProject(Project): Boolean

- **Paramètres :** Project de type Project contenant les nouvelles informations à mettre à jour pour un projet existant.
- **Retourne :** Boolean indiquant le succès ou l'échec de la mise à jour.
- **Utilité :** Permet de mettre à jour les informations d'un projet existant, comme la modification de la description du projet, la mise à jour des dates de début et de fin, etc.

deleteProject(Integer): Boolean

- **Paramètres :** Integer représentant l'identifiant du projet à supprimer.
- **Retourne :** Boolean indiquant le succès ou l'échec de la suppression.
- **Utilité :** Utilisé pour supprimer un projet de la base de données, généralement à la demande de l'utilisateur ou pour des raisons administratives.
- **Interfaces fournies (<<realize>>):**

IProjectManagement

- **Méthodes :**

  createProject(projectDetails: ProjectDetails): Project

  - **Paramètres :** projectDetails de type ProjectDetails contenant les informations nécessaires pour créer un nouveau projet.
  - **Retourne :** Un objet Project représentant le projet nouvellement créé.
  - **Utilité :** Permet aux utilisateurs de créer un nouveau projet dans le système en fournissant des détails tels que le nom du projet, la description, les dates de début et de fin, etc.

  saveProject(project: Project): Boolean

  - **Paramètres :** project de type Project représentant le projet à sauvegarder.
  - **Retourne :** Boolean indiquant le succès ou l'échec de la sauvegarde du projet.
  - **Utilité :** Permet de sauvegarder les modifications apportées à un projet existant, garantissant que les dernières informations sont stockées dans la base de données.

  updateProject(Project): Boolean

  - **Paramètres :** Project de type Project contenant les nouvelles informations à mettre à jour pour un projet existant.
  - **Retourne :** Boolean indiquant le succès ou l'échec de la mise à jour du projet.
  - **Utilité :** Permet aux utilisateurs de mettre à jour les informations d'un projet existant, telles que la modification de la description, la mise à jour des dates, etc.

  deleteProject(projectId: Integer): Boolean

  - **Paramètres :** projectId de type Integer pour l'identifiant du projet à supprimer.
  - **Retourne :** Boolean indiquant le succès ou l'échec de la suppression du projet.

- **Utilité :** Permet aux utilisateurs de supprimer un projet existant du système, généralement lorsqu'un projet est terminé ou annulé.

3. Conception

Interfaces requises (<<use>>):

IDatabasesAccess (spécifique à la conception)

- **Méthodes :**

saveDesign(projectId: Integer, design: Design): Boolean

- **Paramètres :** projectId de type Integer pour l'identifiant du projet associé, et design de type Design pour le design à sauvegarder.
- **Retourne :** Boolean indiquant le succès ou l'échec de la sauvegarde du design.
- **Utilité :** Utilisé pour sauvegarder un design spécifique à un projet dans la base de données, permettant sa récupération et sa révision ultérieures.

loadDesign(projectId: Integer): Design

- **Paramètres :** projectId de type Integer pour l'identifiant du projet dont le design doit être chargé.
- **Retourne :** Un objet Design contenant les informations du design chargé.
- **Utilité :** Permet de charger un design existant depuis la base de données pour un projet donné, facilitant sa visualisation ou sa modification.

updateDesign(projectId: Integer, design: Design): Boolean

- **Paramètres :** projectId de type Integer pour l'identifiant du projet, et design de type Design pour le nouveau design à mettre à jour.
- **Retourne :** Boolean indiquant le succès ou l'échec de la mise à jour du design.
- **Utilité :** Utilisé pour mettre à jour les informations d'un design existant dans la base de données, permettant de refléter les modifications apportées au design.

- **Interfaces fournies (<<realize>>):**

IDesignerTools

- **Méthodes :**

  addDesignElement(element: Element, coord: Coordinates): Boolean

  - **Paramètres :** element de type Element pour l'élément de design à ajouter, et coord de type Coordinates pour les coordonnées où l'élément doit être placé.
  - **Retourne :** Boolean indiquant le succès ou l'échec de l'ajout de l'élément de design.
  - **Utilité :** Permet aux concepteurs d'ajouter de nouveaux éléments de design à un projet, enrichissant ainsi la conception globale.

  removeDesignElement(elementId: Integer): Boolean

  - **Paramètres :** elementId de type Integer pour l'identifiant de l'élément de design à supprimer.
  - **Retourne :** Boolean indiquant le succès ou l'échec de la suppression de l'élément de design.
  - **Utilité :** Permet de supprimer un élément de design spécifique d'un projet, souvent utilisé lors de la révision ou de la modification d'un design.

  updateDesignElement(elementId: Integer, newElement: Element)

  - **Paramètres :** elementId de type Integer pour l'identifiant de l'élément de design à mettre à jour, et newElement de type Element pour le nouvel élément de design.
  - **Retourne :** Boolean indiquant le succès ou l'échec de la mise à jour de l'élément de design.
  - **Utilité :** Utilisé pour modifier les caractéristiques d'un élément de design existant, telles que sa taille, sa couleur, ou sa position, améliorant ainsi la conception globale du projet.

IDesigneViewer

- **Méthodes :**

viewDesign(Integer): Design

- **Paramètres :** Integer pour l'identifiant du design à visualiser.
- **Retourne :** Un objet Design représentant le design à visualiser.
- **Utilité :** Permet aux utilisateurs de visualiser un design spécifique, fournissant une représentation graphique des éléments de design et de leur agencement.

generate3DView(Integer): 2DModel

- **Paramètres :** Integer pour l'identifiant du design à partir duquel générer une vue 3D.
- **Retourne :** Un objet 2DModel représentant la vue 3D générée du design.
- **Utilité :** Offre une fonctionnalité permettant de générer une vue 3D d'un design, améliorant ainsi la compréhension spatiale et la visualisation du projet final.

4. Bibliothèque d'Elements

Interfaces requises (<<use>>):

IDatabaseAccess (spécifique à la bibliothèque d'éléments)

- **Méthodes :**

queryElement(elementId: Element): Boolean

- **Paramètres :** elementId de type Element pour l'identifiant de l'élément à récupérer.
- **Retourne :** Boolean indiquant le succès ou l'échec de la récupération de l'élément.
- **Utilité :** Utilisé pour récupérer les informations d'un élément spécifique de la bibliothèque d'éléments, facilitant ainsi l'accès aux détails de l'élément pour une utilisation ou une évaluation ultérieures.

insertElement(element: Integer): Boolean

- **Paramètres :** element de type Integer pour l'identifiant de l'élément à ajouter dans la bibliothèque.
- **Retourne :** Boolean indiquant le succès ou l'échec de l'ajout de l'élément dans la bibliothèque.
- **Utilité :** Permet d'ajouter de nouveaux éléments dans la bibliothèque d'éléments, enrichissant ainsi les ressources disponibles pour les projets de conception.

updateElement(elementId: Integer, newElement: Element): Boolean

- **Paramètres :** elementId de type Integer pour l'identifiant de l'élément à mettre à jour, et newElement de type Element pour les nouvelles informations de l'élément.
- **Retourne :** Boolean indiquant le succès ou l'échec de la mise à jour de l'élément.
- **Utilité :** Utilisé pour mettre à jour les informations d'un élément existant dans la bibliothèque, permettant de maintenir à jour les détails et les caractéristiques des éléments disponibles.

deleteElement(Integer): Boolean

- **Paramètres :** Integer pour l'identifiant de l'élément à supprimer de la bibliothèque.
- **Retourne :** Boolean indiquant le succès ou l'échec de la suppression de l'élément.
- **Utilité :** Permet de supprimer des éléments obsolètes ou inutiles de la bibliothèque, garantissant que la bibliothèque reste pertinente et à jour.
- **Interfaces fournies (<<realize>>):**

IElementCatalog

- **Méthodes :**

addElement(element: Element): Boolean

- **Paramètres :** element de type Element pour l'élément à ajouter dans la bibliothèque.
- **Retourne :** Boolean indiquant le succès ou l'échec de l'ajout de l'élément.

- **Utilité :** Permet aux utilisateurs d'ajouter de nouveaux éléments dans la bibliothèque d'éléments, augmentant ainsi la variété et la richesse des ressources disponibles pour les projets de conception.

removeElement(elementId: Integer): Boolean

- **Paramètres :** elementId de type Integer pour l'identifiant de l'élément à supprimer de la bibliothèque.
- **Retourne :** Boolean indiquant le succès ou l'échec de la suppression de l'élément.
- **Utilité :** Permet aux utilisateurs de supprimer des éléments spécifiques de la bibliothèque, garantissant que seuls les éléments pertinents et utiles restent disponibles pour une utilisation future.

updateElement(elementId: Integer, newElement: Element): Boolean

- **Paramètres :** elementId de type Integer pour l'identifiant de l'élément à mettre à jour, et newElement de type Element pour les nouvelles informations de l'élément.
- **Retourne :** Boolean indiquant le succès ou l'échec de la mise à jour de l'élément.
- **Utilité :** Permet aux utilisateurs de mettre à jour les informations et les caractéristiques des éléments existants dans la bibliothèque, assurant que les détails des éléments restent précis et à jour.

getElementDetails(elementId: Integer): ElementDetails

- **Paramètres :** elementId de type Integer pour l'identifiant de l'élément dont les détails sont demandés.
- **Retourne :** Un objet ElementDetails contenant les informations détaillées de l'élément spécifié.
- **Utilité :** Permet aux utilisateurs de récupérer les détails complets d'un élément spécifique de la bibliothèque, facilitant l'évaluation de l'élément pour une utilisation potentielle dans un projet de conception.

searchElements(String): Element

- **Paramètres :** String pour les critères de recherche utilisés pour trouver des éléments dans la bibliothèque.
- **Retourne :** Un objet Element représentant l'élément trouvé qui correspond aux critères de recherche.
- **Utilité :** Permet aux utilisateurs de rechercher des éléments dans la bibliothèque en utilisant des critères spécifiques, facilitant ainsi la découverte d'éléments qui pourraient être utilisés dans leurs projets de conception.

getElementCost(Integer): float

- **Paramètres :** Integer pour l'identifiant de l'élément dont le coût est demandé.
- **Retourne :** Un float représentant le coût de l'élément spécifié.
- **Utilité :** Fournit aux utilisateurs le coût d'un élément spécifique de la bibliothèque, aidant ainsi à la planification budgétaire et à la prise de décision pour l'inclusion d'éléments dans un projet de conception.

listElementCosts(): List<Float>

- **Paramètres :** Aucun paramètre requis.
- **Retourne :** Une List<Float> contenant les coûts de tous les éléments de la bibliothèque.
- **Utilité :** Offre une vue d'ensemble des coûts associés à chaque élément de la bibliothèque, permettant aux utilisateurs de prendre des décisions éclairées lors de la sélection d'éléments pour leurs projets de conception.
- Ces descriptions détaillées fournissent un aperçu complet des fonctionnalités et des interactions entre les composants et leurs interfaces respectives, mettant en évidence la manière dont les méthodes sont utilisées pour réaliser diverses opérations au sein du système.

Après avoir exploré en détail la structure interne du système à travers les diagrammes de classe et de composants, ainsi que les interactions entre les objets du système à travers les scénarios et les diagrammes de séquence, il est désormais pertinent de se

concentrer sur les invariants. Les invariants agissent comme des gardiens de l'intégrité et de la cohérence du système, en définissant les règles et les contraintes qui doivent être respectées à tout moment pendant son fonctionnement.

En effet, les invariants viennent compléter la vision statique offerte par les diagrammes de classe et de composants ainsi que la dynamique présentée par les scénarios et les diagrammes de séquence. Ils représentent les conditions sous-jacentes qui garantissent que le système opère de manière fiable et conforme aux attentes, même lorsque celui-ci est soumis à des interactions complexes et variées.

Ainsi, en intégrant les invariants dans notre analyse, nous sommes en mesure d'assurer que le système maintient son intégrité structurelle et fonctionnelle tout au long de son cycle de vie, contribuant ainsi à une expérience utilisateur robuste et cohérente.

VI. Invariants :

A) Les Associations entre les classes :

1. User - Project :

- Invariant : Chaque Project doit avoir un User associé, et un User peut gérer plusieurs Project ou aucun.
- En MOAL : forall p in Project, exists! u in User | u.projects.contains(p)

2. Project - Kitchen :

- Invariant : La relation entre Project et Kitchen est toujours de 1 à 1, garantissant qu'un projet ne peut avoir qu'une seule cuisine et vice versa.
- En MOAL : forall p in Project, exists! k in Kitchen | p.kitchen = k and k.project = p

3. **Kitchen - Element :**

- Invariant : Une Kitchen peut exister sans Element, mais un Element ne peut exister sans être associé à une Kitchen.
- En MOAL : forall e in Element, exists! k in Kitchen | k.elements.contains(e)

4. **Project - Invoice :**

- Invariant : La relation entre Project et Invoice est de 1 à 1, assurant qu'à chaque projet correspond une unique facture.
- En MOAL : forall p in Project, exists! i in Invoice | p.invoice = i and i.project = p

5. **Kitchen - Dimensions et Element - Dimensions :**

- Invariant : Kitchen et Element doivent avoir leurs propres Dimensions définies, garantissant que chaque entité a une taille spécifique.
- En MOAL pour Kitchen : forall k in Kitchen, exists! d in Dimensions | k.dimensions = d
- En MOAL pour Element : forall e in Element, exists! d in Dimensions | e.dimensions = d

6. **ElementLibraryManager - Element :**

- Invariant : ElementLibraryManager peut exister sans Element, mais chaque Element doit être inclus dans une bibliothèque gérée par un ElementLibraryManager.
- En MOAL : forall e in Element, exists elm in ElementLibraryManager | elm.elements.contains(e)

7. **Administrator - User :**

- Invariant : Un Administrator peut exister sans superviser de User, mais chaque User doit être associé à un Administrator.
- En MOAL : forall u in User, exists a in Administrator | a.users.contains(u)

B) L’utilisation de l’application :

1. Unicité de l'Identifiant Utilisateur et de l'E-mail :

- Invariant : Un utilisateur ne peut pas créer un compte avec un identifiant utilisateur (userID) ou un e-mail (email) déjà utilisé par un autre compte. Cela garantit l'unicité de chaque compte utilisateur dans le système.
- En MOAL : forall User u1, User u2 : u1 != u2 ⇒ (u1.email != u2.email and u1.userID != u2.userID)

2. Intégrité de la Connexion :

- Invariant : Un utilisateur doit être authentifié (via login) avant de pouvoir créer un projet (createProject), modifier son profil (updateProfile), ou effectuer toute autre action significative au sein de l'application. Cela assure que seuls les utilisateurs authentifiés peuvent interagir avec l'application.
- En MOAL : forall action in {createProject, updateProfile, ...} : exists User u : u.isAuthenticated()

3. Sécurité du Mot de Passe :

- Invariant : Le mot de passe (password) fourni par l'utilisateur lors de la création d'un compte (createUser) ou de la mise à jour de son profil (updateProfile) doit suivre une politique de sécurité définie (complexité, longueur, etc.), garantissant la sécurité des comptes utilisateurs.
- En MOAL : forall User u : u.setPassword(pw) ⇒ pw.meetsSecurityPolicy()

4. Validité des Dimensions de l'Espace et des Éléments :

- Invariant : Les dimensions (Dimensions) entrées pour une cuisine (Kitchen) ou un élément (Element) doivent être positives et réalistes, assurant la faisabilité de la conception de la cuisine.
- En MOAL pour Kitchen : forall Kitchen k : k.dimensions areValid()
- En MOAL pour Element : forall Element e : e.dimensions areValid()

5. **Gestion des Éléments dans un Projet :**

- Invariant : Les éléments ajoutés à un projet (addElement) doivent exister dans la bibliothèque d'éléments gérée par ElementLibraryManager, garantissant que seuls les éléments disponibles peuvent être utilisés dans les projets.
- En MOAL : forall Project p, Element e : p.addElement(e) ⇒ ElementLibraryManager.contains(e)

6. **Suppression et Mise à Jour Sécurisées :**

- Invariant : La suppression (deleteProject, removeElement) ou la mise à jour (updateElement) d'un projet ou d'un élément ne peut se faire que si l'entité concernée existe déjà et appartient à l'utilisateur effectuant l'action, assurant ainsi l'intégrité et la propriété des données.
- En MOAL pour la suppression : forall User u, Project p : u.deleteProject(p) ⇒ p belongsTo u
- En MOAL pour la mise à jour : forall User u, Element e : u.updateElement(e) ⇒ e belongsTo u

7. **Génération de Facture et Calcul des Coûts :**

- Invariant : Une facture (Invoice) ne peut être générée (generateInvoice) que pour un projet ayant au moins un élément, et le coût total (calculateCost) doit refléter la somme des coûts de tous les éléments inclus dans le projet.
- En MOAL : forall Project p : p.generateInvoice() ⇒ p.elements.size() > 0

8. **Placement Unique des Éléments dans la Cuisine :**

- Invariant : Un emplacement dans une cuisine ne peut contenir qu'un seul élément à la fois, évitant ainsi le chevauchement des éléments.
- En MOAL : pour tout élément e1 et e2 dans Element, si e1.position = e2.position alors e1 = e2, garantissant l'unicité de l'emplacement des éléments.

9. **Conformité des Éléments aux Dimensions de la Cuisine :**

- Invariant : Les éléments placés dans une cuisine ne doivent pas dépasser les dimensions totales de cette dernière, assurant que chaque élément s'intègre correctement dans l'espace alloué.
- En MOAL : pour tout élément e dans Element, pour toute cuisine k dans Kitchen, e.position + e.dimensions <= k.dimensions, ce qui garantit que l'élément est contenu dans les limites de la cuisine.

10. Restriction de Placement Basée sur l'Espace Disponible :

- Invariant : Un élément ne peut être ajouté à la cuisine que si l'espace requis pour cet élément est disponible, sans empiéter sur l'espace occupé par d'autres éléments.
- En MOAL : pour tout élément e ajouté à une cuisine k, not k.elements.any(lambda existing_e: existing_e.occupiesSpace(e.position, e.dimensions)), assurant ainsi qu'il y a suffisamment d'espace libre pour l'élément ajouté.

MY KITCHEN

FLAVOR YOUR SPACE !