ALEXANDRE DEL FABBRO  
 CÉDRIC GARAND  
 FRANCIS PAYAN  
 CLÉMENT PROVENCHER

**Projet Intégrateur 1**  
420-5B5-VI



**DOCUMENT D’ANALYSE ET DE CONCEPTION**

Code Gen

Travail présenté à  
Alexandre Ouellet

Département des Techniques de l’informatique  
 Cégep de Victoriaville  
 Date de remise (Le 20 décembre 2023)

Table des matières

[Introduction 2](#_Toc1609771341)

[Portée du projet 3](#_Toc466845497)

[Étude de faisabilité 3](#_Toc926647588)

[Technologique 4](#_Toc107715060)

[Économique 4](#_Toc1521058045)

[Légal 4](#_Toc1680394614)

[Opérationnel 4](#_Toc1236610547)

[Temporel 4](#_Toc305119363)

[Analyse 4](#_Toc2005676814)

[Acteurs et personas 5](#_Toc1413982788)

[Cas d’utilisation 6](#_Toc914772588)

[Description des cas d’utilisation 7](#_Toc1104866405)

[Cas d’utilisations détaillés 7](#_Toc360911189)

[CU-02-IX SUPPRIMER UNE CLASSE 8](#_Toc862987251)

[CU-01-I CRÉER UN NOUVEAU PROJET FICHE DESCRIPTIVE 9](#_Toc1072101336)

[CU-02-XIII EXPORTER une classe 11](#_Toc1194633438)

[CU-02-VIII Modifier une Méthode 15](#_Toc1340240640)

[CU-02-I AJOUTER une classe 17](#_Toc1490536300)

[CU-02-XIV – Modifier caractéristiques de classe 20](#_Toc911685069)

[CU-02-V ajouter un héritage 22](#_Toc1841477338)

[Plan de test 26](#_Toc1155680715)

# Introduction

Timmy Fortran, un programmeur avec une expérience significative, a sollicité notre équipe pour développer une solution innovante de création de squelettes de code. Notre groupe de quatre développeurs expérimentés a été choisi pour mener à bien ce projet.  
  
La solution que nous développons permettra aux programmeurs de concevoir facilement, via un formulaire intuitif, des structures de code complètes. Ces structures incluront des classes avec attributs, méthodes, et constructeurs, ainsi que des interfaces avec méthodes abstraites, tout en assurant une gestion appropriée de l’héritage. Le système offrira la possibilité d’exporter les résultats en Kotlin et en C#.

Ce document de conception et d'analyse détaille la mise en œuvre de ce projet.

Ce document de conception et d'analyse détaille la mise en œuvre de ce projet. Il commence par explorer les différents scénarios d'utilisation du système, suivi par la structure globale du projet. La section finale décrit les approches de travail adoptées par notre équipe.

# Portée du projet

La solution sert à faciliter et accélérer le développement de code en C# ou Kotlin. Elle permet donc au programmeur de créer des projets puis de générer des squelettes de codes à partir de formulaires. Chaque projet doit être pouvoir être exporté en C# ou en Kotlin.

De plus, le système doit pouvoir enregistrer les projets pour une récupération ultérieure.

**DÉFINITIONS et ACRONYMES**

CU Cas d’utilisation

# Étude de faisabilité

## Technologique

Le logiciel de génération et d’exportation de code en Kotlin et C# comporte peu de contraintes technologiques. Le logiciel utilisera des fichiers locaux pour enregistrer des données d’une session à l’autre.

## Économique

Il n’y a pas d’implication économique puisque les développeurs ne sont pas rémunérés et il n’y a pas d’autres coûts de développement, pas de serveur, pas d’hébergement nécessaire.

## Légal

Au niveau légal, aucune information d’utilisateur n’est gardée donc il n’y a pas de souci de ce côté. La génération de squelette de code est complètement légale. Nous ne sommes pas responsables du code généré par les utilisateurs.

## Opérationnel

**Les opérations du système sont organisées autour de la génération de squelettes de code. Les utilisateurs du logiciel doivent pouvoir :**

1. Gérer plusieurs projets en parallèle
2. Personnaliser des classes
3. Gérer l’héritage

**Ensuite, le système doit offrir les fonctionnalités suivantes :**

1. La sérialisation des données
2. La génération de code selon les informations enregistrées

## Temporel

**Le projet a pour but d’être déployé avant la fin de l’année 2023. Le chemin critique de ce système va comme suit :**

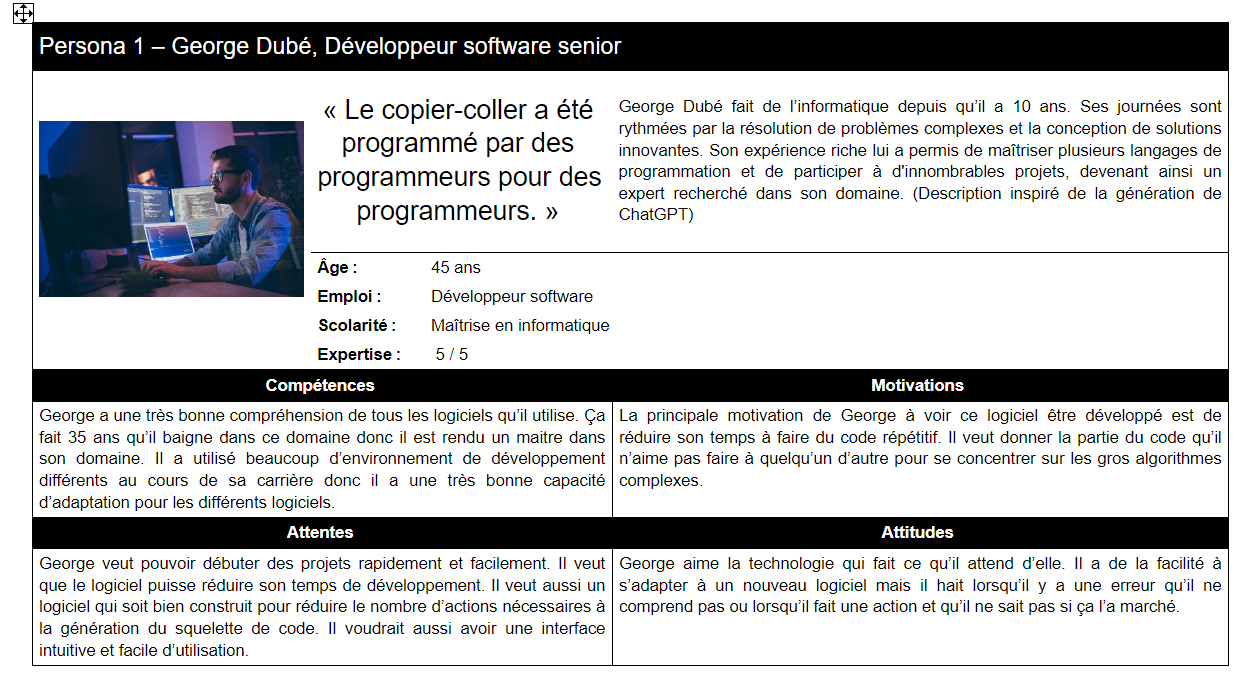
1. Pouvoir créer une classe / interface
2. Remplir les classes / interface avec leurs composants respectifs
3. Exporter le squelette de code

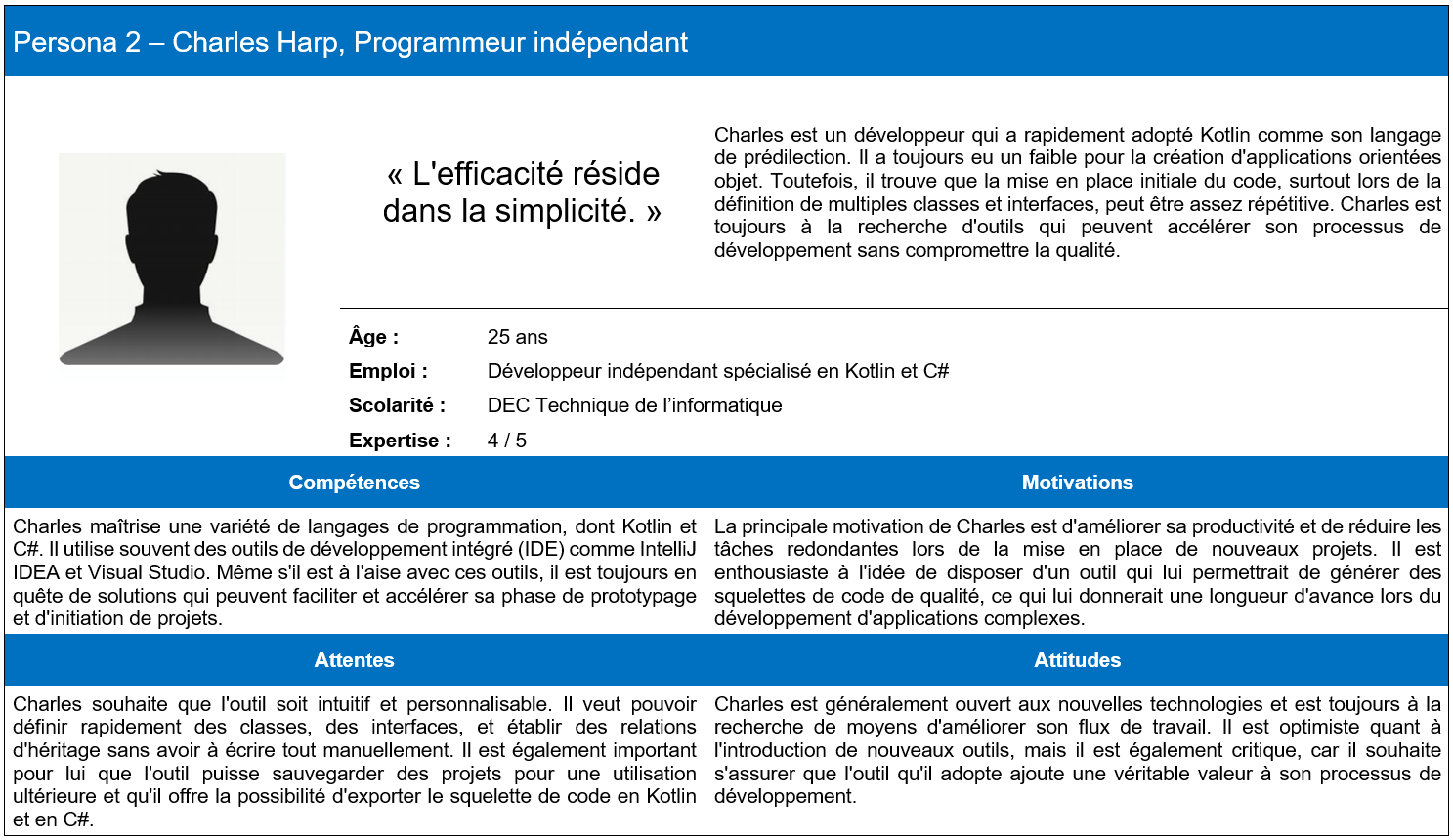
# Analyse

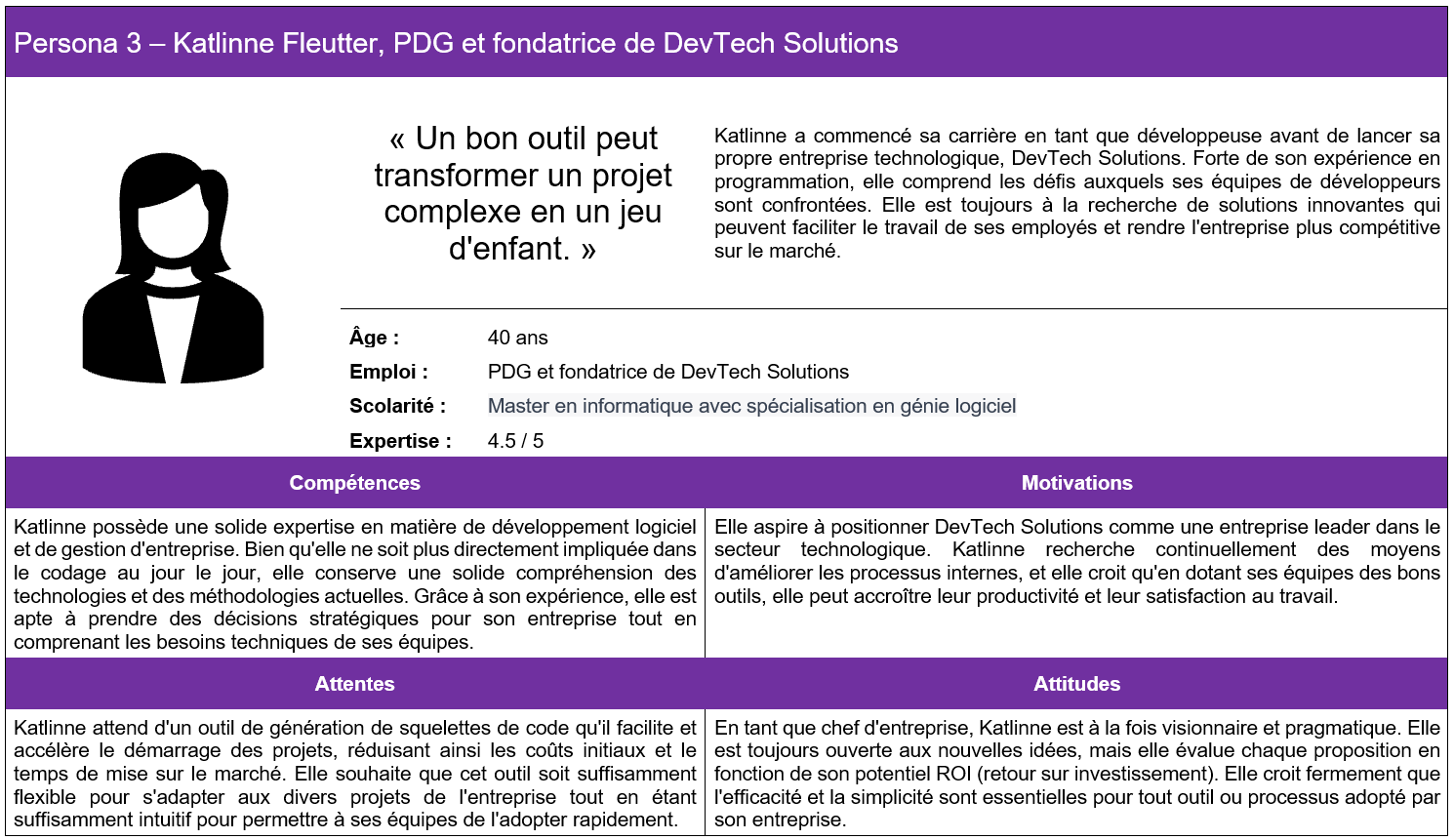
La section d’analyse présente les interactions des entités actrices de la solution.

## Acteurs et personas

|  |  |  |
| --- | --- | --- |
| Entité actrice | Niveau d’opération | Description |
| Programmeur | Primaire | La personne qui utilisera le système pour générer les squelettes de codes |
| Acheteur | Partie prenante | La personne qui achètera la licence de l’application |





\* Idées de Personas réalisés à l’aide de Chat-GPT [*https://chat.openai.com/chat*](https://chat.openai.com/chat)

# Cas d’utilisation

## Description des cas d’utilisation

**Liste des cas d’utilisation**

|  |  |  |
| --- | --- | --- |
| **Numéro du cas** | **Nom du cas** | **Personne actrice principale** |
| **CU-01** | **Gérer un projet** | **Programmeur** |
| CU-01-I | Créer un nouveau projet | **Programmeur** |
| CU-01-II | Ouvrir un projet | **Programmeur** |
| CU-01-III | Sélectionner une classe | **Programmeur** |
| CU-01-IV | Sélectionner une interface | **Programmeur** |
| CU-01-V | Exporter un projet | **Programmeur** |
| CU-01-VI | Changer le chemin du dossier d’exportation | **Programmeur** |
| **CU-02** | **Gérer une classe** | **Programmeur** |
| CU-02-I | Ajouter une classe | **Programmeur** |
| CU-02-II | Ajouter une propriété | **Programmeur** |
| CU-02-III | Ajouter un constructeur | **Programmeur** |
| CU-02-IV | Ajouter une méthode | **Programmeur** |
| CU-02-V | Ajouter un héritage | **Programmeur** |
| CU-02-VI | Modifier une classe (nom) | **Programmeur** |
| CU-02-VII | Modifier propriétés constructeur | **Programmeur** |
| CU-02-VIII | Modifier méthode | **Programmeur** |
| CU-02-IX | Supprimer une classe | **Programmeur** |
| CU-02-X | Supprimer une propriété | **Programmeur** |
| CU-02-XI | Supprimer méthode | **Programmeur** |
| CU-02-XII | Supprimer un héritage | **Programmeur** |
| CU-02-XIII | Exporter une classe | **Programmeur** |
| CU-02-XIV | Modifier caractéristiques de classe | **Programmeur** |
| **CU-03** | **Gérer une interface** | **Programmeur** |
| CU-03-I | Ajouter une interface | **Programmeur** |
| CU-03-II | Modifier une interface (nom) | **Programmeur** |
| CU-03-III | Supprimer une interface | **Programmeur** |
| CU-03-IV | Implémenter une interface | **Programmeur** |
| CU-03-V | Enlever une implémentation | **Programmeur** |
| CU-03-VI | Exporter une interface | **Programmeur** |
| CU-03-VII | Modifier caractéristiques d’interface | **Programmeur** |

Cas d’utilisations détaillés

### CU-02-IX SUPPRIMER UNE CLASSE

#### **Fiche descriptive**

**Code :** CU-02-IX   
**Nom :** Supprimer une classe  
**Introduit** **:** 0.0.1  
**Auteur :** **Francis Payan**

#### **Entités actrices**

**Primaire :** Programmeur

#### **Déclencheur**

1. Le programmeur sélectionne la classe à supprimer et appuie sur « Supprimer ».

#### **Préconditions**

1. Une classe est sélectionnée.

#### **Postconditions**

1. La classe est supprimée, ainsi que les dépendances liées à l'héritage si confirmées par l'utilisateur.

#### **Scénario nominal**

|  |  |  |
| --- | --- | --- |
| **Étape** | **Action de la personne actrice principale** | **Réponse du système** |
| 1 | Le programmeur sélectionne la classe à supprimer et appuie sur « Supprimer ». | Le système vérifie si la classe a des dépendances liées à l'héritage.    Si la classe est une classe "parent", une indication visuelle, comme une icône ou une marque spécifique, apparaît à côté de la classe pour informer discrètement l’utilisateur que d'autres classes héritent de cette classe. |
| 2 | Le programmeur confirme ou annule la suppression. | Si confirmé, le système supprime la classe et la dépendance d'héritage des classes enfants. Sinon, il annule l'opération. |

#### **Scénario d’extenSion**

1a. Le développeur souhaite seulement supprimer la classe parente, mais pas les classes enfants.

1. Le système propose de transformer les classes enfants en classes autonomes, éliminant la dépendance d'héritage, sans supprimer ces classes.

#### **Scénario d’exception**

1a. Une erreur survient lors de la suppression de la classe ou des classes associées.

1. Le système affiche le message d'erreur « Une erreur est survenue lors de la suppression. Veuillez réessayer ultérieurement. »

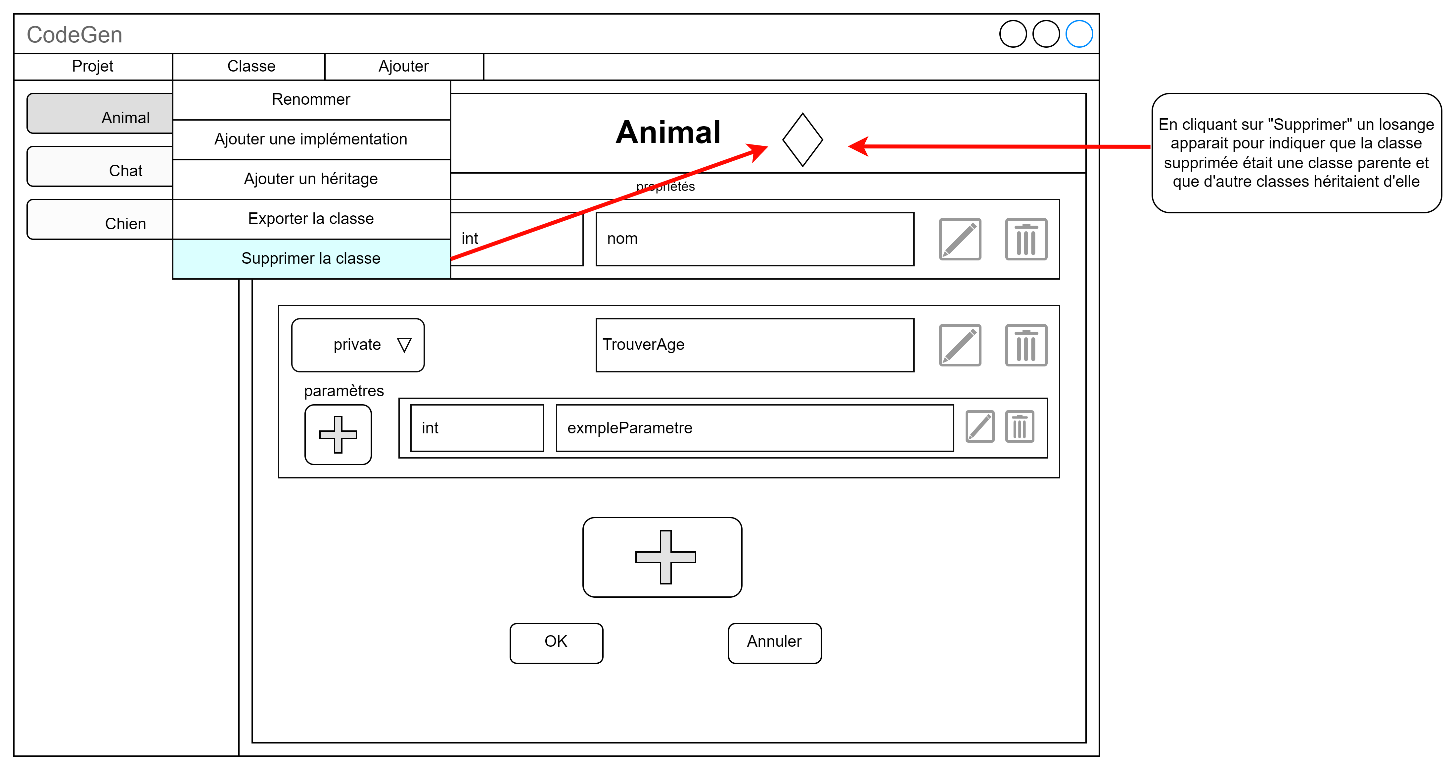
2a. La classe sélectionnée est utilisée dans d'autres modules ou dépendances non liées à l'héritage.

1. Le système informe le développeur des autres dépendances impliquées. Le système remplace la référence de la classe supprimée par un type spécial « Élément Manquant ». Cela permet de garder une trace des références qui étaient liées à la classe supprimée. Les éléments marqués comme « Élément Manquant » seront clairement identifiables dans le code et dans toute exportation, alertant ainsi le développeur de la nécessité de réviser ces références.

#### **Spécifications non fonctionnelles**

1. L’interface est actualisée après la suppression de la classe.

#### **Prototype d’écran**



### CU-01-I CRÉER UN NOUVEAU PROJET FICHE DESCRIPTIVE

Fiche descriptive

**Code :** CU-01-I   
**Nom :** Créer un nouveau projet  
**Introduit** **:** 0.0.1  
**Auteur :** **Francis Payan**

#### **Entités actrices**

**Primaire :** Programmeur

#### **Déclencheur**

1. Le programmeur clique sur le bouton « Créer un nouveau projet ».

#### **Préconditions**

1. Le programmeur à le droit d’accès au système.

#### **Postconditions**

1. Le projet est correctement créé et enregistré par le système.

#### **Scénario nominal**

|  |  |  |
| --- | --- | --- |
| **Étape** | **Action de la personne actrice principale** | **Réponse du système** |
| 1 | Le programmeur saisit un nom pour le nouveau projet. | Le système vérifie que le nom du projet respecte la convention de nommage et qu'il est unique dans le système. |
| 2 | Le programmeur appuie sur «Enregistrer». | Le système crée le projet et dirige le programmeur vers la page de formulaire de squelette de code avec le nom du projet affiché en haut de la hiérarchie. |

#### **Scénarios d’extension**

1a. Le programmeur a besoin des directives pour le nommage du projet.

1. Le programmeur clique sur le point d'interrogation dans le coin supérieur droit du popup.
2. Le système affiche une fenêtre contextuelle ou une infobulle avec des informations détaillées sur les règles de nommage du projet, des exemples de noms valides et des conseils pour choisir un nom de projet pertinent. Exemple d’informations :
   1. Une explication de la longueur minimale et maximale du nom.
   2. Des informations sur les caractères spéciaux autorisés ou interdits.
   3. Des conseils pour créer un nom de projet descriptif et pertinent.
   4. Des instructions pour éviter les noms de projet déjà utilisés dans le système.

#### **Scénarios d’exception**

1a. Le nom du projet est invalide ou manquant.

1. Le programmeur tente d'enregistrer un nouveau projet sans entrer un nom ou avec un nom qui ne suit pas les règles de nommage établies.
2. Le système affiche le message d’erreur : « Le nom du projet est requis et doit suivre les conventions de nommage. Cliquez sur le point d'interrogation pour obtenir de l'aide. »
3. Le bouton d’enregistrement reste désactivé tant que le champ du nom n'est pas correctement rempli.

1b. Le nom du projet existe déjà dans le système.

1. Le programmeur entre un nom de projet déjà utilisé dans le système et tente de l'enregistrer.
2. Le système affiche le message d’erreur : « Ce nom de projet est déjà utilisé. Veuillez en choisir un autre ou cliquer sur le point d'interrogation pour des suggestions. »
3. Le bouton d’enregistrement reste désactivé tant que le nom du projet n'est pas modifié.

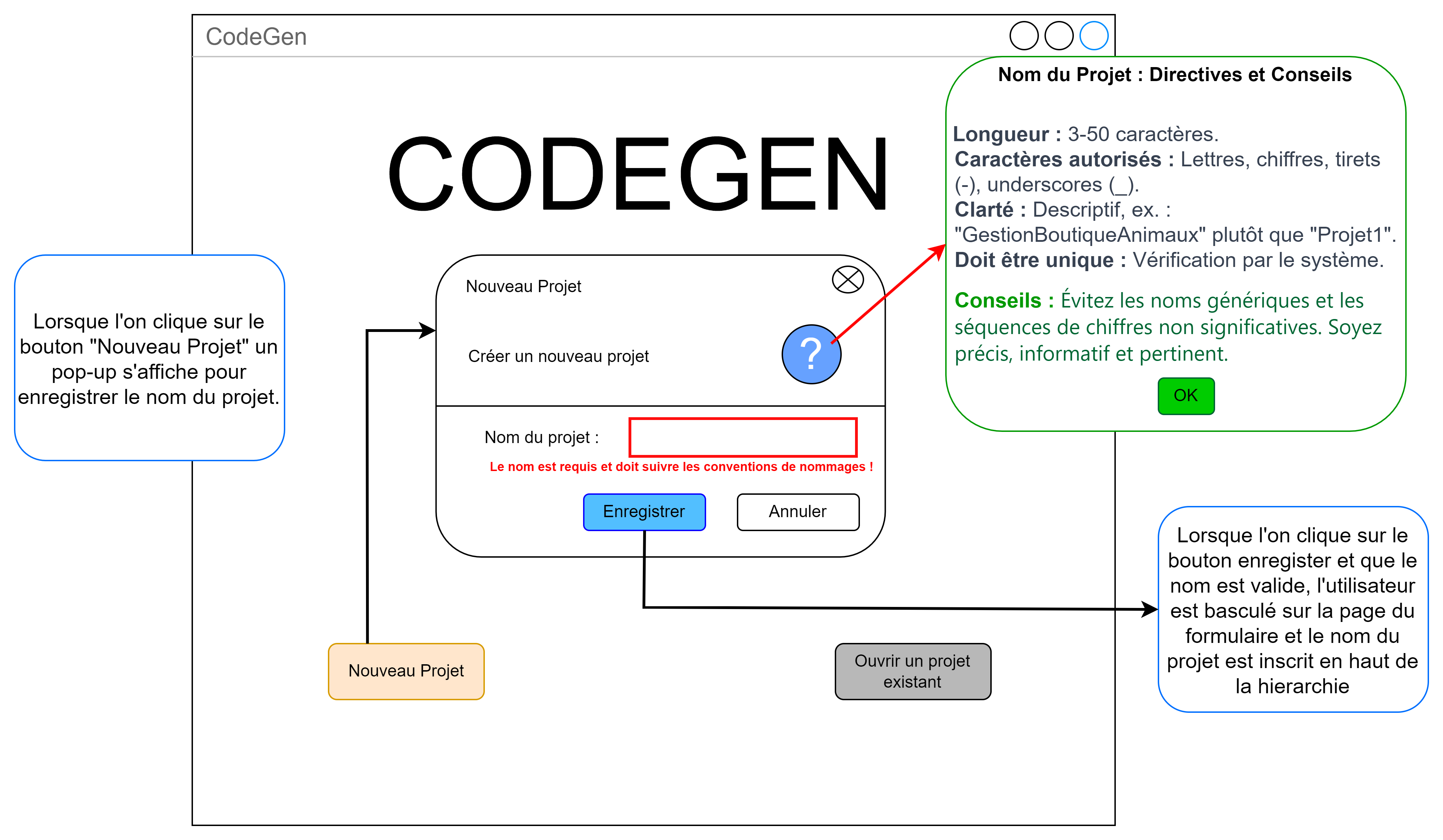
1c. Erreur système lors de la tentative de création du projet.

1. Une erreur système survient empêchant la création du projet après que le programmeur ait appuyé sur « Enregistrer ».
2. Le système affiche le message d’erreur : « Une erreur est survenue lors de la création du projet. Veuillez réessayer ou contacter le support si le problème persiste. »

#### **Spécifications non fonctionnelles**

* Les messages d'erreur doivent être affichés en temps réel, immédiatement après que l'action ayant causé l'erreur a été effectuée par l'utilisateur.
* Les informations fournies après avoir cliqué sur le point d'interrogation doivent être claires, concises et facilement compréhensibles.

#### **Prototype d’écran**



### CU-02-XIII EXPORTER une classe

#### **Fiche descriptive**

**Code :** CU-02-XIII  
**Nom :** Exporter une classe   
**Introduit** **:** 0.0.1  
**Auteur :** **Cédric Garand**

#### **Acteurs**

**Primaire : Programmeur**

#### **Déclencheur**

1. Le programmeur appuie sur le bouton « Exporter la classe » dans l’onglet « Classe ».

#### **Préconditions**

1. Une classe doit être sélectionnée.

#### **Postconditions**

1. Un fichier de code contenant le code de la classe est généré et téléchargé.

#### **Scénario nominal**

|  |  |  |
| --- | --- | --- |
| **Étape** | **Action de la personne actrice principale** | **Réponse du système** |
| 1 | Le programmeur appuie sur le bouton « Exporter la classe » dans l’onglet « Classe ». | Le système affiche une fenêtre contextuelle qui demande à l'utilisateur s’il veut exporter la classe en C# ou en Kotlin. Le chemin où le fichier va être enregistré est affiché à l’écran |
| 2 | Le programmeur sélectionne une des deux options. | Le système ferme la fenêtre contextuelle de choix de langage. Le système valide si les types inscrits font partie de la liste de types. Le système compile les éléments que l'utilisateur a mis dans sa classe et génère un fichier contenant le squelette du code généré. Le fichier est téléchargé dans l’ordinateur de l’utilisateur à l’endroit où il l’a défini. Le système affiche une fenêtre contextuelle avec un message de succès. |
| 3 | Le programmeur appuie sur le bouton « OK » | Le système ferme la fenêtre contextuelle. |

#### **Scénarios d’extension**

1a. Le programmeur appuie sur le bouton pour modifier le chemin d’exportation

1. Il sélectionne le chemin d’exportation où le fichier va être enregistré
2. Le nouveau chemin est affiché à l’écran et va seulement être utilisé pour cette exportation
3. Le reste du scénario nominal se poursuit

2a. Un ou plusieurs types inscrits dans la classe ne font pas partie de la liste.

1. Le système affiche une fenêtre contextuelle d'avertissement qui indique que les types n’ont pas été vérifié donc le code généré pourrait ne pas fonctionner.
2. Le programmeur choisi s’il veut continuer ou annuler.
3. Si le programmeur choisi de continuer, le fichier de code est généré puis téléchargé et le système remplace la fenêtre contextuelle d’avertissement avec une fenêtre contextuelle avec un message de succès. Le scénario nominal se poursuit à l’étape 3. Si le programmeur choisi d’annuler, le système ferme la fenêtre contextuelle d’avertissement et le fichier de code n’est pas téléchargé.

#### **Scénarios d’exception**

2a. Une erreur survient lors de la compilation.

1. Le système affiche une fenêtre contextuelle avec le message d’erreur suivant : « L’exportation n’a pas réussi dû à une erreur du système ».
2. Le programmeur appuie sur le bouton « OK », ce qui ferme la fenêtre contextuelle. Le téléchargement ne s’effectue pas.

2b. Le chemin du dossier d’exportation n’est pas valide

1. Le système affiche une fenêtre contextuelle avec le message d’erreur suivant : « Le chemin du dossier d’exportation n’est pas valide ».
2. Le programmeur appuie sur le bouton « OK », ce qui ferme la fenêtre contextuelle. Le téléchargement ne s’effectue pas.

#### **Spécifications non fonctionnelles**

* Les fenêtres contextuelles peuvent être soumises en appuyant sur la touche « Retour »
* Tous les contrôles sont désactivés lorsque les fenêtres contextuelles sont affichées.

#### **Prototype d’écran**

Une image contenant texte, capture d’écran, nombre, affichage

Description générée automatiquement

Une image contenant texte, diagramme, ligne, Parallèle

Description générée automatiquement

### CU-02-VIII Modifier une Méthode

#### **Fiche descriptive**

**Code :** CU-02-VIII  
**Nom :** Modifier une méthode  
**Introduit** **:** 0.0.2  
**Auteur :** **Cédric Garand**

#### **Acteurs**

**Primaire : Programmeur**

#### **Déclencheur**

1. Le programmeur appuie sur l’icône de crayon à la droite du nom de la méthode.

#### **Préconditions**

1. Une classe doit être sélectionnée.
2. Une méthode doit être ajouté dans la classe

#### **Postconditions**

1. Le nom de la méthode et/ou son type de retour et/ou sa visibilité sont modifiés

#### **Scénario nominal**

|  |  |  |
| --- | --- | --- |
| **Étape** | **Action de la personne actrice principale** | **Réponse du système** |
| 1 | Le programmeur appuie sur l’icône de crayon à la droite du nom de la méthode | Le système débloque les champs textes pour le nom et le type de retour de la méthode et la combo-box pour la visibilité. |
| 2 | Le programmeur modifie le nom de la méthode et/ou le type de retour de la méthode et/ou la visibilité de la méthode. Lorsqu’il a fini de modifier ce qu’il veut, le programmeur appui sur l’icône de crochet pour enregistrer les informations. | Le système enregistre les nouvelles informations et bloque les champs textes et le combo-box pour empêcher la modification. |

#### **Scénarios d’exception**

2a. Le nom de la méthode et/ou le type de retour ont plus que 255 caractères

1. Le système affiche un message d’erreur en rouge sous le champ qui est en erreur indiquant que le nombre de caractères ne peut pas être au-dessus de 255. La modification n’est pas exécutée.

2b. Le nom de la méthode et/ou le type de retour sont vides

1. Le système affiche un message d’erreur en rouge sous le champ qui est en erreur indiquant que le champ ne peut pas être vide. La modification n’est pas exécutée.

#### **Spécifications non fonctionnelles**

* Les champs de texte sont grisés pour indiquer qu’ils ne peuvent pas être modifiées

#### **Prototype d’écran**

Une image contenant texte, capture d’écran, diagramme, Parallèle

Description générée automatiquement

Une image contenant texte, diagramme, capture d’écran, ligne

Description générée automatiquement

### CU-02-I AJOUTER une classe

**FICHE DESCRIPTIVE**

**Code :** CU-02-I   
**Nom :** Ajouter une classe   
**Introduit** **:** 0.0.1   
**Auteur :** **Clément Provencher**

**ACTEURS**

**Primaire : Programmeur**

**DÉCLENCHEUR**

1. Le programmeur appuie sur le bouton « Classe » dans l’onglet « Ajouter »

**PRÉCONDITIONS**

1. Un projet doit être ouvert
2. Aucune autre fenêtre contextuelle ne doit être ouverte

**POSTCONDITIONS**

1. Une classe doit être ajoutée à la liste de classes dans la barre de gauche, et la nouvelle classe doit se sélectionner automatiquement
2. Si la classe ajoutée implémente des interfaces, les méthodes de l’interface sont déjà ajoutées

**SCÉNARIO NOMINAL**

|  |  |  |
| --- | --- | --- |
| **Étape** | **Action de la personne actrice principale** | **Réponse du système** |
| 1 | Le programmeur appuie sur le bouton « Classe » dans l’onglet « Ajouter ». | Le système affiche une fenêtre contextuelle pour l’ajout d’une nouvelle classe |
| 2 | Le programmeur entre le nom de la classe dans le champ prévu à cet effet | Le champs affiche le nom que le programmeur est en train d’entrer |
| 3 | Le programmeur appuie sur le bouton « Ajouter la classe » | La classe est générée puis ajoutée à la liste de classe à gauche de l’écran.  La fenêtre contextuelle se ferme ensuite et la nouvelle classe est sélectionnée |

**SCÉNARIOS D’EXTENSION**

2a. Le programmeur clique sur le bouton « Ajouter un héritage »

1. Ceci lance le **CU-02-V**
2. La classe héritée est affichée sous le bouton précédemment cliqué pour indiquer quelle classe est héritée en ce moment

2b. Le programmeur clique sur le bouton « Ajouter une implémentation »

1. Ceci lance le **CU-03-I**
2. Les interfaces implémentés (il peut il y en avoir plus qu’une) sont affichées sous le bouton précédemment cliqué pour indiquer quelles interfaces sont implémentées en ce moment

**SCÉNARIOS D’EXCEPTION**

1a. Le nom de la classe n’a pas été entré

1. L’ajout de la classe est empêché
2. Un message sous le champ indique au programmeur qu’il doit nommer la classe

**SPÉCIFICATIONS NON FONCTIONNELLES**

* Le nom de la classe ne peut pas dépasser 255 caractères

#### **Prototype d’écran**

Une image contenant texte, capture d’écran, diagramme, Parallèle

Description générée automatiquement

Une image contenant texte, capture d’écran, diagramme, Parallèle

Description générée automatiquement

# CU-02-XIV – Modifier caractéristiques de classe

**FICHE DESCRIPTIVE**

**Code :** CU-02-XIV   
**Nom :** Modifier caractéristiques de classe   
**Introduit** **:** 0.0.2   
**Auteur :** **Clément Provencher**

**ACTEURS**

**Primaire : Programmeur**

**DÉCLENCHEUR**

N/A

Un déclencheur n’existe pas pour ce cas d’utilisation, car il est actif tant et aussi longtemps que la classe demeure sélectionnée.

**PRÉCONDITIONS**

1. Un projet doit être ouvert
2. Une classe doit être sélectionnée

**SCÉNARIO NOMINAL**

|  |  |  |
| --- | --- | --- |
| **Étape** | **Action de la personne actrice principale** | **Réponse du système** |
| 1 | Le programmeur appuie sur la boîte combinée de visibilité | Le système ouvre la liste des choix de visibilités |
| 2 | Le programmeur sélectionne la visibilité qui lui convient | La liste de choix de visibilités est fermée et la boîte combinée affiche maintenant la visibilité sélectionnée |
|  |  |  |
| 1 | Le programmeur appuie sur la boîte combinée de mot-clé de classe | Le système ouvre la liste des choix de mots-clés |
| 2 | Le programmeur sélectionne le mot-clé qui lui convient | La liste de choix de mots-clés est fermée et la boîte combinée affiche maintenant le mot-clé sélectionné |

Les deux combinaisons d’étapes (étapes 1 et 2) sont optionnelles et peuvent être réaliser dans n’importe quel ordre (en respectant l’ordre interne, 1 doit précéder 2)

**SPÉCIFICATIONS NON FONCTIONNELLES**

* La classe ne peut avoir d’autres visibilités que public, private et protected
* La classe ne peut avoir d’autre mots-clés que static, sealed et abstract

#### **Prototype d’écran**

Une image contenant texte, capture d’écran, diagramme, Parallèle

Description générée automatiquement

### CU-02-V ajouter un héritage

#### **FICHE DESCRIPTIVE**

**Code :** CU-02-V  
**Nom :** Ajouter un héritage  
**Introduit :** 0.0.1  
**Auteur :** Alexandre del Fabbro

#### **ENTITÉ ACTRICE**

**Primaire :** Personne qui programme

#### **DÉCLENCHEURs**

1. La personne qui programme appuie sur « Ajouter un héritage » dans l’onglet « Classe ».
2. La personne qui programme appuie sur “Ajouter un héritage” lors de la création d’une nouvelle classe.

#### **préconditions**

1. Le formulaire ouvert doit représenter une classe.

OU

1. Une nouvelle classe est en création.

#### **postconditions**

1. L’héritage est ajouté à la classe.
2. Les méthodes abstraites choisies sont implémentées.

#### **Scénario nominal**

|  |  |  |
| --- | --- | --- |
| **Étape** | **Action de la personne actrice principale** | **Réponse du système** |
| 1 | La personne qui programme appuie sur le bouton « Ajouter un héritage » | Le système fait apparaitre la fenêtre de sélection d’un héritage. |
| 2 | La personne qui programme sélectionne un héritage dans la liste. | Le système vérifie que l’héritage de la classe sélectionné est possible et ne constitue pas de boucle avec d’autres classes. Le système active le bouton d’ajout et affiche la liste des méthodes abstraites pouvant être implémentées, l’utilisateur peut cocher celles qu’il veut ajouter. |
| 3 | La personne qui programme appuie sur le bouton d’ajout. | Le système désactive la fenêtre. Les méthodes abstraites du parent sont implémentées dans leurs parties respectives du formulaire. |

## 

#### **scénarios d’extension**

1a. La classe possède déjà un héritage.

1. Le message « Une classe ne peut posséder qu’un seul héritage. » devient rouge et le message « L’héritage actuel sera retiré si un nouvel héritage est implémenté et ses méthodes retirées. » est affiché.
2. Reprise à l’étape 2.
3. Le système remplace l’héritage original par le nouveau, enlève les méthodes héritées du premier et implémente les méthodes abstraites du nouvel héritage.

#### **scénarios d’exception**

2a. La classe choisie est *sealed*.

1. Le système affiche le message « La classe [nom de la classe] est *sealed*. Son implémentation peut provoquer des problèmes à l’exportation. ».
2. Reprise à l’étape 3.

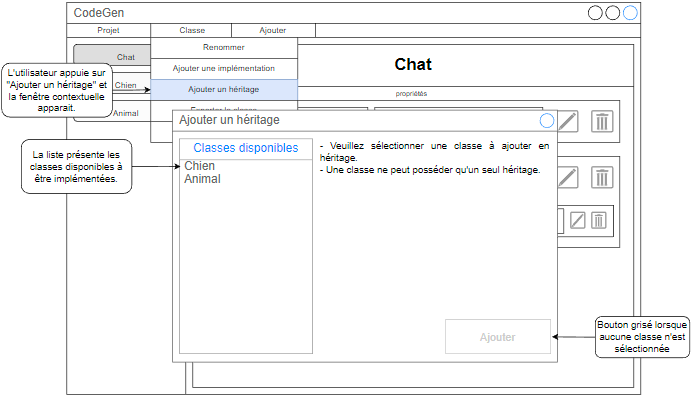
2b. La classe choisie cause un héritage circulaire.

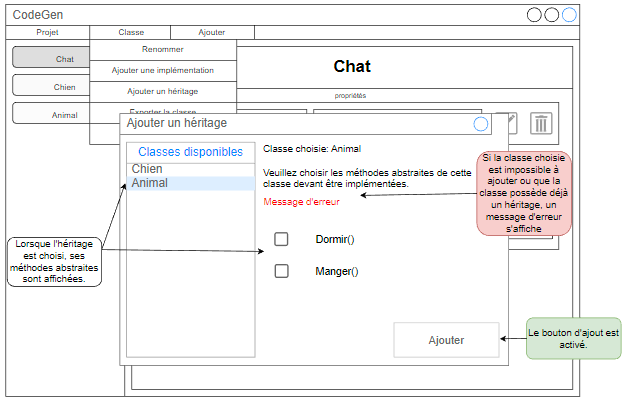
1. Le système affiche le message « Cet héritage conduit à un héritage circulaire. Son implémentation est impossible ».
2. Le bouton de confirmation n’est pas activé.
3. La liste des méthodes abstraites n’est pas affichée.

#### **spécifications non fonctionnelles**

* Utiliser la fenêtre pour lister toutes les classes.
* Sélectionner une classe pour lister les méthodes abstraites.
* L’ouverture de la fenêtre contextuelle désactive le reste de l’application.

#### **Prototype d’écran**





### CU-01-II Ouvrir un projet

#### **FICHE DESCRIPTIVE**

**Code :**  CU-01-II  
**Nom :**  Ouvrir un projet  
**Introduit :**  0.0.1  
**Auteur :** Alexandre del Fabbro

#### **ENTITÉ ACTRICE**

**Primaire :** Personne qui programme

#### **DÉCLENCHEURs**

1. Lorsque le programme s’ouvre, la personne qui programme appuie sur “Ouvrir un projet” dans la fenêtre contextuelle qui apparait.
2. La personne qui programme appuis sur “Ouvrir un projet” dans le menu “Projet” de l’interface principale.

#### **préconditions**

1. Un projet doit exister.

#### **postconditions**

1. L'interface principale présente le projet ouvert.

#### **Scénario nominal**

|  |  |  |
| --- | --- | --- |
| **Étape** | **Action de la personne actrice principale** | **Réponse du système** |
| 1 | La personne qui programme appuie sur le bouton "Ouvrir un projet”. | Le système fait apparaitre la fenêtre contextuelle de sélection d’un projet existant. |
| 2 | La personne qui programme sélectionne un projet dans la liste. | Le système vérifie que le projet est sain et peut être ouvert. Le système active le bouton de confirmation. |
| 3 | La personne qui programme appuie sur le bouton de confirmation. | Le système désactive la fenêtre contextuelle. Le projet est affiché dans l’interface principal. |

## 

#### **Scénarios D’EXCEPTION**

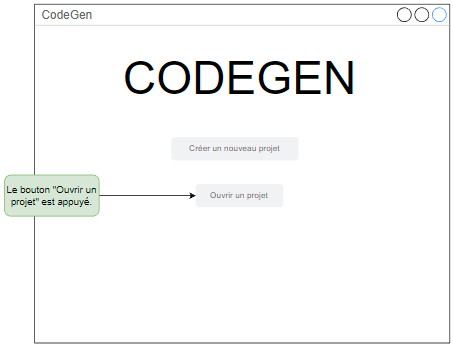
2a. Le projet choisi ne peut pas être ouvert.

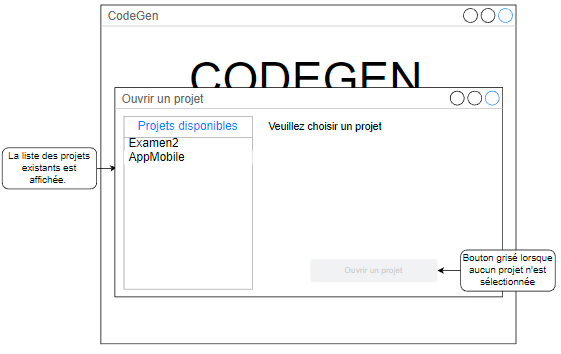
1. Le système affiche le message "Un problème est survenu avec le projet choisi. Veuillez consulter le fichier des projets ou choisir un autre projet”.
2. Reprise à l’étape 2

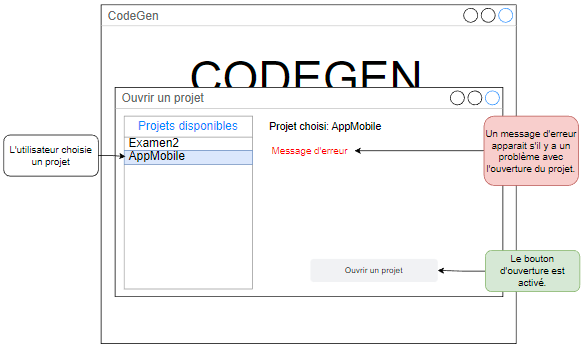
#### **spécifications non fonctionnelles**

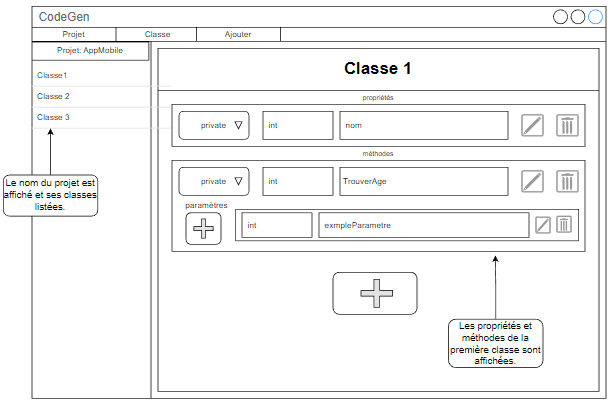
* Utiliser la fenêtre pour lister tous les projets.
* Utiliser le choix de projet pour tester si le fonctionnement est correct.
* L’ouverture de la fenêtre contextuelle désactive le reste de l’application.

#### **Prototype d’écran**









# Plan de test

La stratégie de test pour ce système va comme suit :

Des tests unitaires seront effectués pour toutes les méthodes de base des composants qui sont la fondation même du programme. Il est important de les tester, car ils constituent la base de tout le reste du programme. Il est donc impératif que tout fonctionne correctement.

De plus, des tests d'intégration seront effectués pour vérifier le bon fonctionnement de la sérialisation des données. Des tests d'intégration seront également réalisés pour valider le bon fonctionnement de l'exportation.

Des tests de charge semblent inutiles, puisque notre système ne fait appel à aucun serveur. Il n'y a rien à surcharger.

