

Ministry of Higher Education, Research and Innovation

**THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE**

**Speciality:** Computer Science and Entrepreneurship

|  |
| --- |
| **design and development of a mobile tontine management application.** |

Thesis’ ID : ……………..

**President:** First name Last name,

Grade

**Member:** First name Last name,

Grade,

**Supervised by**: First name Last name

Title

**Under the supervision by:** First name Last name

Title (Teacher, Researcher, Dr ...)

Presented and publicly defended:

**Armel OUEDRAOGO**

**Academic year: 2022 – 2023**

À ma famille

# Acknowledgements

Je tiens à exprimer ma profonde gratitude envers BIT (Burkina Institute of Technology) pour m'avoir offert l'opportunité d'étudier et de poursuivre mes aspirations académiques. Les ressources, le soutien et l'environnement d'apprentissage stimulant que vous avez fourni ont grandement contribué à ma croissance personnelle et à ma réussite.

Je souhaite également remercier sincèrement Monsieur Roland TOUGMAN, mon directeur de mémoire, pour son engagement, son expertise et son soutien inestimable tout au long de mon travail de recherche. Ses conseils éclairés, ses encouragements constants et sa disponibilité ont été d'une importance capitale pour moi. Grâce à sa guidance, j'ai pu approfondir mes connaissances et développer des compétences précieuses.

Enfin, j'aimerais adresser mes remerciements à Monsieur le directeur du BIT pour son leadership éclairé, sa vision inspirante et son engagement envers l'excellence académique. Sa détermination à fournir un environnement d'apprentissage de qualité et à encourager la réussite des étudiants a été une source de motivation pour moi.

# Abstract

# Resumé

# TABLE OF CONTENT

[Acknowledgements II](#_Toc139265588)

[Abstract III](#_Toc139265589)

[Resumé IV](#_Toc139265590)

[TABLE OF CONTENT V](#_Toc139265591)

[List of Acronyms AND Abbreviations. VII](#_Toc139265592)

[LIST OF TABLES VIII](#_Toc139265593)

[LIST OF FIG IX](#_Toc139265594)

[Introduction 1](#_Toc139265595)

[Part I: Theoretical studies 3](#_Toc139265596)

[Chap 1: General information on the tontine 3](#_Toc139265597)

[Introduction 3](#_Toc139265598)

[I Contexte, problématique et objectifs 3](#_Toc139265599)

[I.1 Contexte et problématique 3](#_Toc139265600)

[I.2 Objectifs 4](#_Toc139265601)

[I.2.1 Objectif général 4](#_Toc139265602)

[I.2.2 Objectifs spécifiques 4](#_Toc139265603)

## [II Revue de littérature sur latontine ……………………………………………………….. 5](#_Toc139265604)

[II.1 Etude de l’existant 5](#_Toc139265607)

[II.3 Les défis liés à la tontine 6](#_Toc139265608)

[Conclusion 6](#_Toc139265609)

[Chap 2: Methodologie and conceptual approache 7](#_Toc139265610)

[I Méthodologie de conception 7](#_Toc139265611)

[I.1 Introduction à la méthodologie de conception centrée sur l'utilisateur (CCU) 7](#_Toc139265612)

[I.2 L’importance de la méthodologie CCU 8](#_Toc139265613)

[I.3 Planification du projet avec la méthode CCU 9](#_Toc139265614)

[I.3.1 Collecte des exigences des utilisateurs 9](#_Toc139265615)

[I.3.2 Création de maquettes interactives 9](#_Toc139265616)

[I.3.3 Tests et itérations avec les utilisateurs 9](#_Toc139265617)

[I.3.4 Prototypage et développement de l'application 9](#_Toc139265618)

[II Approche conceptuelle 10](#_Toc139265619)

[II.1 Architecture logicielle 10](#_Toc139265620)

[II.1.1 Architecture 3 tiers 10](#_Toc139265621)

[II.1.2 Avantages de l’architecture 3 tiers 11](#_Toc139265622)

[II.2 Méthode de modélisation 12](#_Toc139265623)

[II.2.1 Diagrammes de classes 13](#_Toc139265624)

[II.2.2 Diagrammes de séquence 14](#_Toc139265625)

[II.2.3 Diagramme de cas d'utilisation 14](#_Toc139265626)

[II.3 Les technologies de conception 15](#_Toc139265627)

[II.3.1 Framework pour le frontend 15](#_Toc139265628)

[II.3.1.1 Flutter 15](#_Toc139265629)

[II.3.1.2 Avantages et inconvénients de Frameworks 16](#_Toc139265630)

[II.3.2 Choix de notre framework 17](#_Toc139265632)

[II.4 Backend 18](#_Toc139265633)

[II.4.1 Firebase 18](#_Toc139265634)

[II.4.2 Avantages & Limites de Firebase 18](#_Toc139265635)

[III Biliographie 19](#_Toc139265636)

# List of Acronyms AND Abbreviations.

|  |  |  |
| --- | --- | --- |
| BIT | **:** | Burkina Institute of Technology |
| CCU | **:** | Conception Centrée sur l’Utilisateur |
| GPU |  | Graphics Processing Unit |
| MVC | **:** | Model View Controller |
| SDK |  | Software Development Kit |
| UML | **:** | Unified Modeling Language |

# LIST OF TABLES

[Tableau 1 Flutter& React Native: Avantages et inconvénients 17](#_Toc139265536)

[Tableau 2 Avantages & Limites de firebase 18](#_Toc139265537)

# LIST OF FIG

[**Figure 1 : UCD process** source : 8](file:///C:\Users\Issiaka%20OUEDRAOGO\Documents\Soutenance%20docs\IssiakaMemory%20version2.docx#_Toc139265550)

[Figure 2 3-tier client-server architecture Source: researchgate. 10](file:///C:\Users\Issiaka%20OUEDRAOGO\Documents\Soutenance%20docs\IssiakaMemory%20version2.docx#_Toc139265551)

[Figure 3 UML diagram structure 13](file:///C:\Users\Issiaka%20OUEDRAOGO\Documents\Soutenance%20docs\IssiakaMemory%20version2.docx#_Toc139265552)

[Figure 4: Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2022 15](file:///C:\Users\Issiaka%20OUEDRAOGO\Documents\Soutenance%20docs\IssiakaMemory%20version2.docx#_Toc139265553)

# Introduction

Les tontines sont des systèmes traditionnels de financement et d'épargne communautaires qui ont perduré à travers les générations. Dans le contexte actuel, où la technologie joue un rôle central dans notre quotidien, concevoir une application dédiée à la gestion de tontine offre de nombreux avantages pour les participants et les administrateurs.

L'objectif de notre projet est de proposer une solution numérique qui facilite la gestion des tontines, en automatisant les processus manuels et en offrant une interface conviviale pour les utilisateurs. Cette application vise à simplifier les tâches liées à la collecte des cotisations, à la gestion des contributions et à la distribution équitable des fonds, tout en assurant la transparence et la sécurité des transactions.

Dans le cadre de notre recherche, nous avons réalisé une revue de la littérature sur les tontines et les applications existantes dans ce domaine. Cette analyse nous a permis de comprendre les principes fondamentaux des tontines, ainsi que les fonctionnalités clés attendues dans une application de gestion de tontine. Nous avons également identifié les lacunes et les défis associés à la gestion traditionnelle des tontines, ce qui nous a motivés à proposer une solution innovante basée sur la technologie.

Notre approche de conception se distingue par son orientation vers les besoins des participants de tontine, en mettant l'accent sur la convivialité et l'accessibilité de l'application. Nous avons également pris en compte les aspects de sécurité des données et de confidentialité des informations sensibles pour garantir la confiance des utilisateurs.

Dans cette étude, nous abordons deux aspects essentiels. Tout d'abord, nous examinons une partie théorique qui présente des informations générales sur la tontine, ainsi que les méthodologies et les approches conceptuelles utilisées pour ce projet Ensuite, nous nous concentrons sur la partie design et implémentation, où nous décrivons en détail le design adopté, la modélisation utilisée et l'approche de conception. De plus, nous présentons les étapes d'implémentation et les résultats obtenus. Enfin, nous discuterons des perspectives d'amélioration et des possibilités d'extension de l'application pour répondre aux besoins spécifiques des différentes communautés et contextes.

En conclusion, ce mémoire a pour objectif de présenter notre recherche sur la conception d'une application de gestion de tontine, offrant une alternative numérique aux systèmes traditionnels de tontine. Nous espérons que notre travail contribuera à faciliter les opérations de tontine, à renforcer la confiance des participants et à promouvoir l'inclusion financière au sein des communautés.

# Part I: Theoretical studies

# Chap 1: General information on

## Introduction

La tontine est une méthode de collecte et de gestion de fonds communautaires qui repose sur la participation régulière des membres et le partage des fonds accumulés. Cette section du chapitre vise à fournir une présentation claire et concise des principes et du fonctionnement de la tontine. Ces informations décrivent entre autres : le contexte, la problématique, les objectifs et se termine par le concept de la tontine

# Contexte, problématique et objectifs

## Contexte

La tontine est une forme ancienne et répandue de coopération financière et sociale qui existe dans de nombreuses régions du monde. Elle repose sur un système de collecte de cotisations régulières de la part d'un groupe de personnes, avec la promesse qu'à tour de rôle, chaque membre recevra une somme d'argent prédéfinie. Les tontines sont souvent utilisées pour financer des projets individuels, faire face à des dépenses imprévues ou épargner pour des besoins futurs.

Historiquement, les tontines ont été gérées de manière informelle, reposant sur la confiance mutuelle des participants et des processus manuels pour la collecte et la distribution des fonds. Cependant, ces approches traditionnelles peuvent présenter des défis majeurs en termes d'efficacité, de transparence et de sécurité.

### Problématique

La problématique de la tontine réside dans les difficultés inhérentes à sa gestion manuelle et informelle. Les processus manuels de collecte des cotisations peuvent être sujets à des erreurs, à des retards et à des oublis. De plus, la répartition des fonds peut soulever des questions de transparence et de confiance, et les litiges peuvent survenir en cas de désaccord ou de déséquilibre dans les paiements.

En outre, les tontines traditionnelles peuvent être limitées en termes de fonctionnalités et de possibilités d'évolution. Les participants peuvent avoir des besoins spécifiques, tels que la gestion de plusieurs tontines, des options d'épargne ou des mécanismes de planification financière, qui ne sont pas toujours pris en compte dans les approches traditionnelles.

Face à ces défis, la conception d'une application de gestion de tontine offre une solution moderne et innovante. Elle vise à automatiser et à simplifier les processus de collecte, de suivi et de distribution des fonds, tout en renforçant la transparence, la sécurité et la convivialité de l'expérience des participants.

La problématique qui se pose est donc de concevoir une application de gestion de tontine qui réponde aux besoins spécifiques des participants, tout en améliorant l'efficacité, la transparence et la sécurité des opérations. Cette application devra être adaptée aux réalités culturelles, sociales et économiques des tontines, tout en exploitant les avancées technologiques pour offrir une expérience utilisateur optimale.

## Objectifs

### Objectif général

L'objectif général de l'application de gestion de tontine mobile est de fournir aux participants une plateforme numérique conviviale et sécurisée pour la gestion efficace et transparente de leurs tontines. L'application vise à automatiser les processus de collecte des cotisations, de suivi des transactions et de répartition des fonds, tout en offrant des fonctionnalités supplémentaires telles que la consultation des soldes, des outils de planification financière et des notifications personnalisées. L'objectif est d'améliorer l'expérience des participants, de renforcer la confiance et la transparence au sein des tontines, et de faciliter la gestion administrative pour les responsables des groupes de tontine. En intégrant la technologie mobile dans la gestion des tontines, l'objectif est de moderniser cette pratique ancestrale et de favoriser une participation plus large et une utilisation plus efficace des ressources financières au sein des communautés.

### Objectifs spécifiques

L'objectif spécifique de l'application de gestion de tontine mobile est de :

* Simplifier la collecte des cotisations : Permettre aux participants de contribuer facilement et de manière sécurisée aux tontines via l'application, en offrant des options de paiement flexibles et en automatisant les rappels de paiement. Concevoir une interface conviviale et intuitive pour l'application mobile de vote électronique, en prenant en compte les différentes contraintes et les préférences des utilisateurs.
* Faciliter le suivi des transactions : Permettre aux participants de consulter en temps réel l'historique de leurs contributions, les montants reçus et les dates de paiement, offrant ainsi une transparence totale des transactions effectuées au sein de la tontine.
* Automatiser la répartition des fonds : Mettre en place des mécanismes automatisés pour garantir une distribution équitable des fonds aux membres de la tontine conformément aux règles établies, réduisant ainsi les risques d'erreurs et de litiges.
* Offrir des fonctionnalités de gestion avancées : Permettre aux administrateurs des tontines de gérer facilement les paramètres de la tontine, d'accéder à des rapports financiers détaillés, de gérer les membres et d'envoyer des notifications importantes à travers l'application.
* Renforcer la sécurité des données : Mettre en place des mesures de sécurité robustes pour protéger les informations sensibles des participants, telles que les données financières et les informations personnelles, en utilisant des protocoles de chiffrement et des pratiques de sécurité de pointe.
* Améliorer l'expérience utilisateur : Concevoir une interface utilisateur conviviale et intuitive, offrant une navigation fluide et une expérience agréable pour les participants de la tontine, quel que soit leur niveau de compétence technologique.

Ces objectifs spécifiques guideront notre travail de recherche et de développement afin d'atteindre l'objectif général.

# . Revue de littérature sur la tontine

La présente étude de cas porte sur la tontine qui est une pratique courante en milieu rural et surtout urbain où le taux de monétarisation est relativement plus important. Ces structures apparaissent aujourd’hui indispensables eu égard à leur contribution dans la mobilisation de l'épargne et la distribution de crédit pour les couches à revenus modestes. Elles suscitent des espoirs réels dans une société où les structures financières modernes répondent encore imparfaitement aux besoins de la grande majorité de la population.[1]



## Etude de l’existant

Au cours de la revue de la littérature, nous avons identifié plusieurs études et projets qui se sont intéressés à l'utilisation de la technologie pour améliorer la gestion des tontines. Parmi les principales applications de gestion de tontine existantes, certaines se distinguent par leurs fonctionnalités innovantes et leur adoption par les utilisateurs. Voici une synthèse des principales conclusions de notre revue de littérature :

**Tontine Trust est** une fintech fondée sur le principe qu'après avoir travaillé et épargné pendant plusieurs décennies, vous devriez maintenant pouvoir vous détendre et profiter de votre retraite sans craindre de manquer d'argent dans les années à venir.[2]

**Tontine rotative** est un système ou les participants d'une tontine s'engagent à verser une somme prédéterminée à une fréquence donnée. Pour chaque tour de versement, un des participants est désigné pour être le bénéficiaire des fonds des autres participants. Deux modes de désignation du bénéficiaire existent : soit il y a tirage au sort avant chaque versement, soit le tirage au sort est fait une seule fois au début du cycle et les participants bénéficient des versements en fonction du numéro qu'ils ont tiré au sort.

Lorsque tous les participants ont été bénéficiaires des fonds une fois, le cycle de la tontine est terminé. À la fin du cycle (quand tous les participants ont déjà reçu les versements) un nouveau cycle est généralement entamé. [3]

**Tontine à accumulation** Dans une tontine à accumulation, les cotisations ne sont pas redistribuées à un des membres mais accumulées dans la caisse de la tontine. Les fonds ainsi collectés appartiennent à la tontine jusqu'à ce que les membres décident d'effectuer un partage, c'est-à-dire de redistribuer tout l'argent accumulé aux membres, au prorata de ce qu'ils ont cotisé. [3]

## Les défis liés à la tontine

La tontine peut présenter plusieurs défis, tant du point de vue de la gestion que de l'expérience des participants. Voici quelques défis courants liés à la tontine :

* **Collecte des cotisations :** L'un des défis majeurs dans une tontine est la collecte régulière des cotisations auprès des participants. Il peut y avoir des retards de paiement, des oublis ou des difficultés à récupérer les contributions, ce qui peut affecter le bon fonctionnement de la tontine.
* **Transparence et confiance :** La tontine repose sur la confiance mutuelle entre les participants. Cependant, il peut y avoir des préoccupations quant à la transparence des transactions, à l'équité de la répartition des fonds ou aux soupçons de comportements frauduleux. La préservation de la confiance au sein du groupe de tontine est essentielle pour assurer son bon fonctionnement.
* **Gestion des litiges :** En cas de désaccord sur la répartition des fonds, les dates de paiement ou d'autres aspects de la tontine, des litiges peuvent survenir entre les participants. La gestion de ces litiges peut être complexe et nécessiter des discussions et des négociations pour parvenir à un consensus.
* **Gestion administrative :** La gestion manuelle des tontines peut être chronophage et sujette à des erreurs. Le suivi des cotisations, le calcul des montants à verser à chaque membre et la tenue des registres peuvent devenir fastidieux, surtout lorsque la tontine compte un grand nombre de participants.
* **Sécurité des fonds et des informations** : La sécurité des fonds et la protection des informations personnelles des participants sont des préoccupations importantes. Les tontines traditionnelles peuvent être vulnérables aux vols, aux pertes ou aux abus, ce qui souligne l'importance d'adopter des mesures de sécurité appropriées.
* **Accès à l'inclusion financière** : Dans certaines régions, l'accès à des services financiers formels peut être limité, et les tontines jouent alors un rôle essentiel pour les individus et les communautés. Cependant, il peut exister des obstacles pour les populations marginalisées ou les groupes défavorisés qui souhaitent participer à une tontine.

## Conclusion

# La conception d'une application de gestion de tontine peut contribuer à relever certains de ces défis en automatisant les processus de collecte des cotisations, en améliorant la transparence des transactions, en facilitant la communication entre les participants et en renforçant la sécurité des fonds et des informations. Cependant, il est essentiel de prendre en compte ces défis lors de la conception et du développement de l'application afin de garantir une solution adaptée aux besoins des participants de la tontine.

# Chap 2: Methodologie and conceptual approache

Le deuxième chapitre de cette première partie se concentre sur la méthodologie et l’approche conceptuelle utilisées pour concevoir et développer l'application mobile de gestion de tontine.

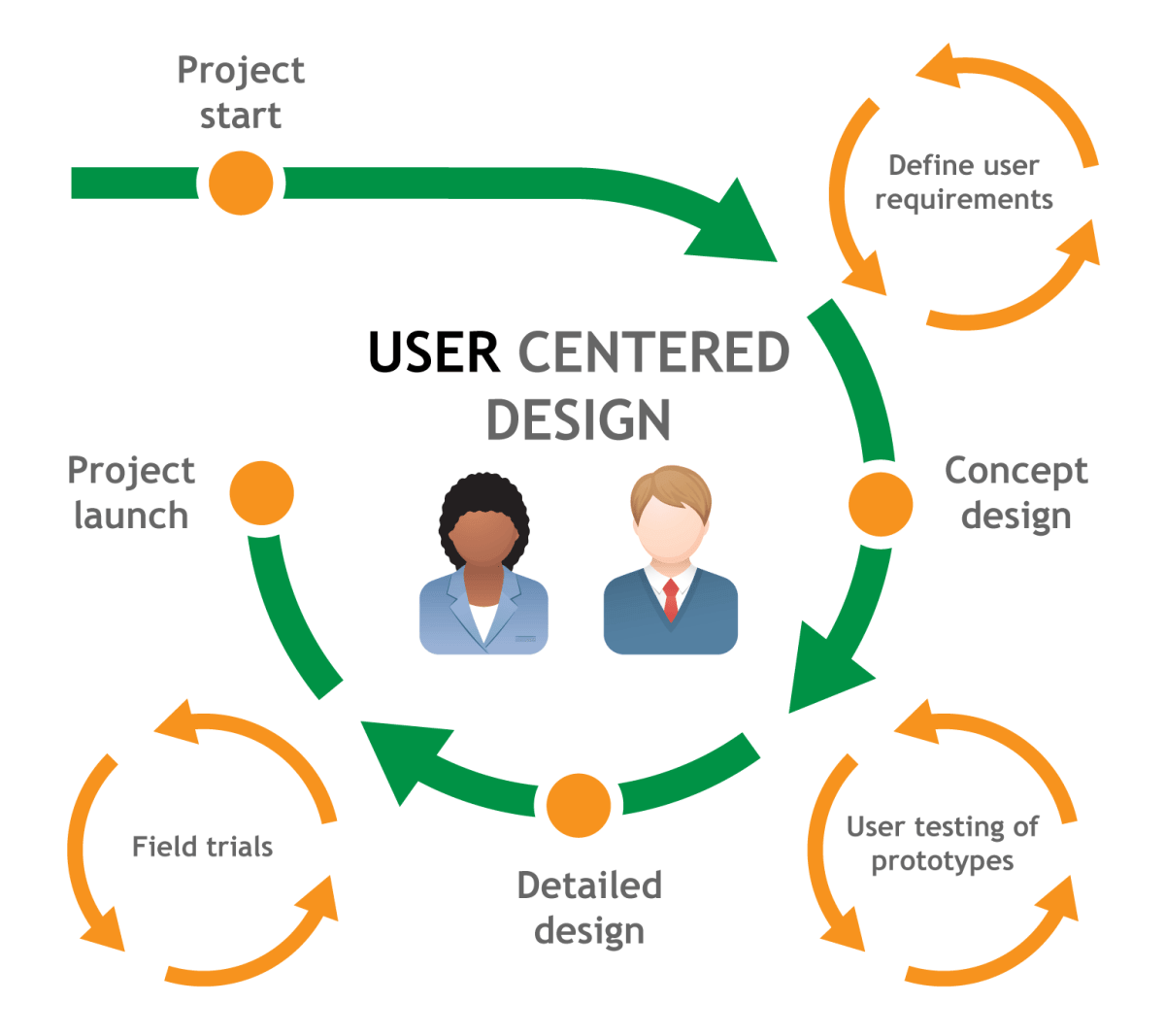
# Méthodologie de conception

## Introduction à la méthodologie de conception centrée sur l'utilisateur (CCU)

Le terme "conception centrée sur l'utilisateur" a été inventé pour la première fois par le théoricien de la conception Donald Norman, et la pratique a commencé à évoluer vers la fin des années 1980. La CCU définit les phases du cycle de vie de la conception ou du développement sur la base d'une compréhension approfondie de QUI utilisera le produit, de ses tâches et de son environnement comme l’indique la **figure1**. La méthodologie de conception centrée sur l'utilisateur est une approche qui place les besoins et les attentes des utilisateurs au cœur du processus de conception d'une application. La CCU est le processus par lequel tout produit, tel que l'interface utilisateur (IU) d'un site web ou d'une application, est conçu du point de vue de la manière dont il sera compris et utilisé par un utilisateur humain.[4]

Elle implique une compréhension approfondie des utilisateurs cibles, de leurs objectifs et de leur contexte d'utilisation.

Plutôt que de forcer les utilisateurs à adapter leurs attitudes et leurs comportements pour utiliser un système, ce dernier devrait être conçu pour soutenir les croyances, les attitudes et les comportements existants des utilisateurs visés. [4]



**Figure 1 : UCD process** source :

## L’importance de la méthodologie CCU

L'objectif de la CCU est de créer des produits qui présentent un degré élevé de convivialité pour l'utilisateur.[7]

En concevant avec une attention constante pour l'utilisateur. Les concepteurs ne sont plus libres de s'exprimer dans leur travail pour quelque raison que ce soit ; au contraire, ils sont obligés de se concentrer sur ce que l'utilisateur aimerait voir ou sur son comportement. [4]

En se concentrant sur les comportements et les objectifs de l'utilisateur, la conception sera plus facile à utiliser et plus accessible. Le résultat d'une conception basée sur la CCU est une expérience utilisateur réussie[4].

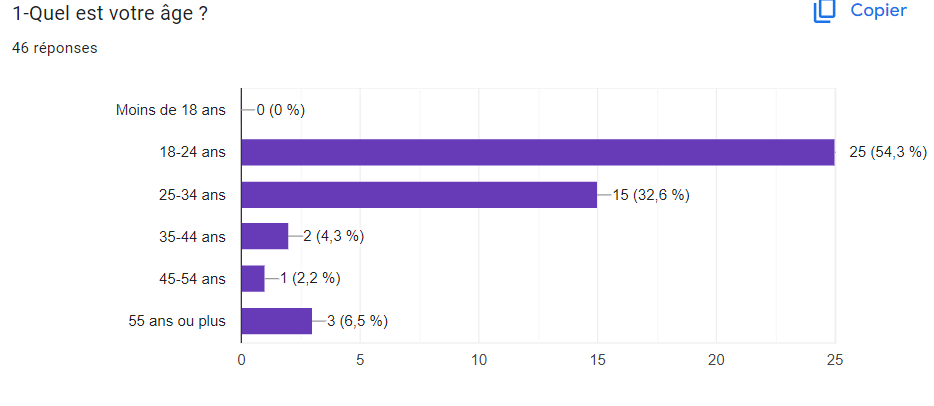
Dans le cadre de notre projet de développement de l'application mobile de gestion de tontine, cette méthodologie nous a permis de créer une expérience utilisateur optimale en répondant aux besoins spécifiques des étudiants.

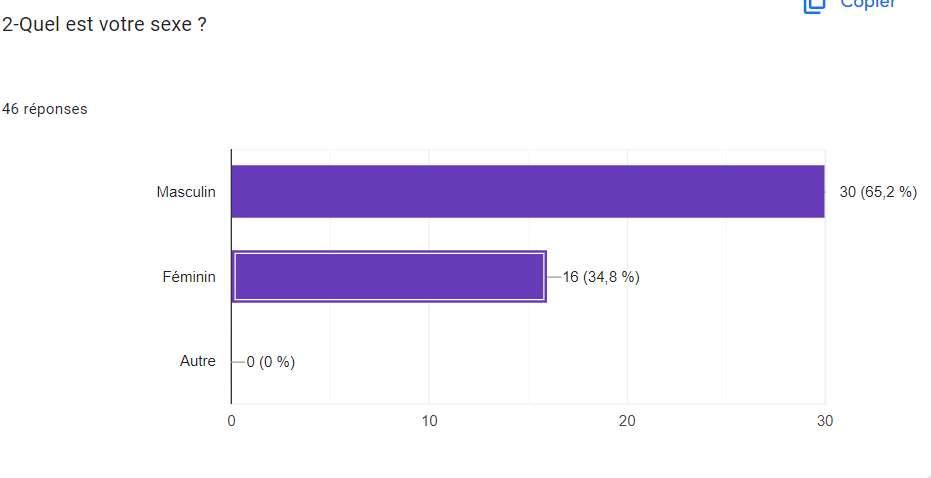
## Planification du projet avec la méthode CCU

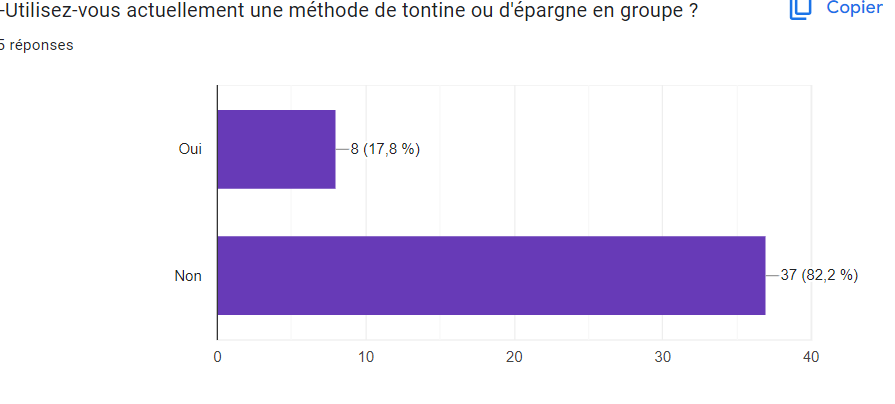
La méthodologie CCU comprends 4 étapes clés dans le développement d’un projet informatique.

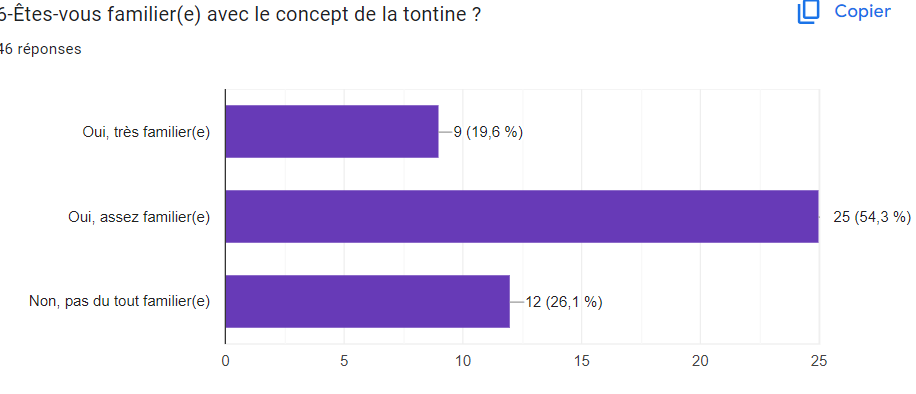
### Collecte des avis des utilisateurs

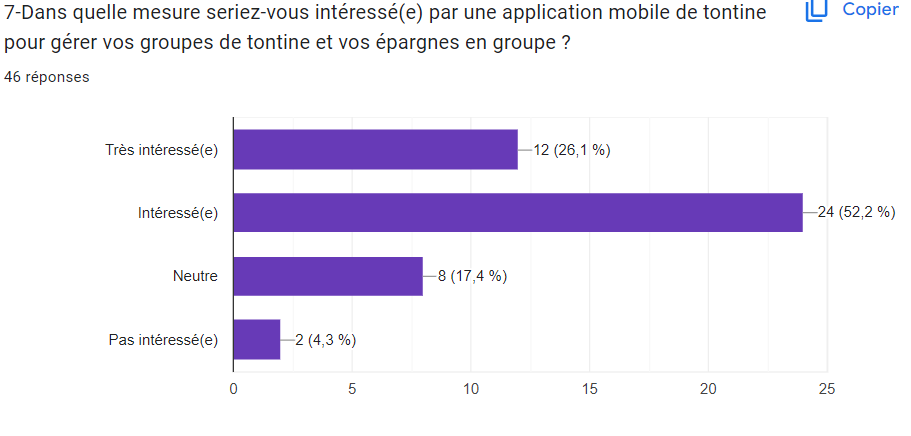
La première étape de notre méthodologie de conception centrée sur l'utilisateur consiste à collecter les avis des utilisateurs sur la mise en place d’une application de gestion de tontine. Nous avons mené des sondages et des observations pour comprendre leurs besoins, leurs préférences et leurs attentes par rapport à l'application.

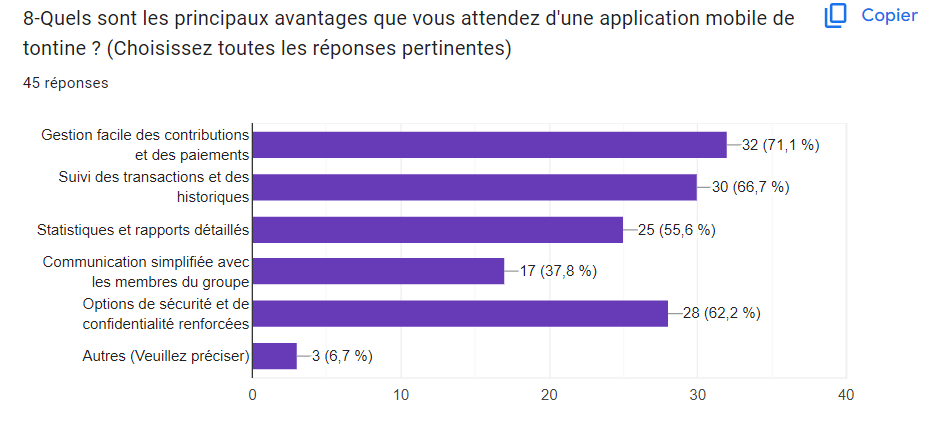












### Création de maquettes interactives

Une fois que nous avons recueilli les exigences des utilisateurs, nous nous sommes passés à la création de maquettes interactives de l'application. Les maquettes sont des représentations visuelles des différentes interfaces et fonctionnalités de l'application. Nous avons utilisé un outil de conception tel qu’Adobe XD pour créer des maquettes interactives qui permettent aux utilisateurs de naviguer à travers l'application comme s'ils l'utilisaient réellement.

### Tests et itérations avec les utilisateurs

Une fois les maquettes créées, nous les avons soumis aux utilisateurs pour obtenir leurs commentaires et leurs retours. Nous avons organisé des séances de test où les utilisateurs ont interagit avec les maquettes et ont exprimé leurs impressions, leurs difficultés et leurs suggestions. Nous avons analysé ensuite ces retours et apporté des modifications aux maquettes en fonction des recommandations des utilisateurs. Ce processus itératif de tests et d'itérations nous a permis d'affiner progressivement la conception de l'application en nous assurant de répondre aux besoins et aux attentes des utilisateurs.

### Prototypage et développement de l'application

Une fois que les maquettes ont été validées par les utilisateurs, nous sommes passés à la phase de prototypage et de développement de l'application. Nous avons utilisé un framework pour développer l'application mobile en nous basant sur les maquettes approuvées. Nous avons procédé par étapes, en implémentant les fonctionnalités clés de l'application et en effectuant des tests réguliers pour garantir la qualité et la performance.

# Approche conceptuelle

## Architecture logicielle

La conception de l'architecture logicielle joue un rôle essentiel dans le développement d’une application mobile de gestion de tontine. Elle permet de définir la structure globale de l'application, en identifiant les différents composants, modules et interactions entre eux. Dans le cadre de notre projet, nous allons présenter notre approche de conception de l'architecture logicielle en se basant sur l’architecture 3 tiers.

### Architecture 3 tiers

L'architecture logicielle 3 tiers, également connue sous le nom d'architecture client-serveur en trois couches, est un modèle de conception couramment utilisé pour développer des applications.

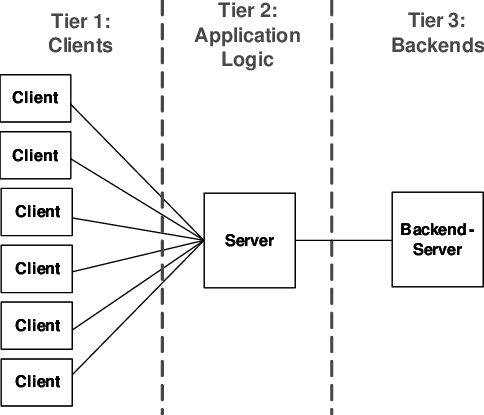


Figure 2 3-tier client-server architecture Source: researchgate.

Elle divise une application en trois couches distinctes comme l’indique la figure2, chacune ayant ses propres responsabilités et rôles bien définis. Ces couches sont :

**La couche de présentation** (ou interface utilisateur) : Cette couche est responsable de l'interaction entre l'utilisateur et l'application. Elle comprend les éléments visuels tels que les écrans, les menus, les boutons et les formulaires. Son objectif principal est de fournir une interface conviviale et réactive à l'utilisateur. Dans cette couche, on retrouve généralement des technologies comme les langages de programmation front-end (par exemple, JavaScript, HTML, CSS) et les frameworks de développement d'interfaces utilisateur (par exemple, Angular, React, Vue.js, flutter).

**La couche logique** (ou couche métier) : Cette couche contient la logique de traitement des données et des fonctionnalités spécifiques de l'application. Elle est responsable de la manipulation des données, des règles métier, des calculs et de la logique d'application. Cette couche traite les demandes provenant de la couche de présentation et effectue les opérations nécessaires pour y répondre. Elle est souvent implémentée à l'aide de langages de programmation back-end (par exemple, Java, C#, Python) et de frameworks applicatifs (par exemple, Spring, ASP.NET, Django).

**La couche de données** : Cette couche est responsable de la gestion des données utilisées par l'application. Elle peut inclure des bases de données relationnelles ou non relationnelles, des services Web, des API ou d'autres sources de données externes. La couche de données est chargée de stocker, récupérer, mettre à jour et supprimer les données nécessaires à l'application. Elle fournit également une interface pour accéder aux données à partir de la couche logique. Des technologies telles que les systèmes de gestion de bases de données (par exemple, MySQL, Oracle, MongoDB) et les API REST sont souvent utilisées dans cette couche.

### Avantages de l’architecture 3 tiers

* Objets persistants légers à haute performance
* Évolutivité - Chaque couche peut se mettre à l'échelle horizontalement
* Performance - Étant donné que la couche de présentation peut mettre en cache les requêtes, l'utilisation du réseau est réduite au minimum, et la charge est allégée sur les couches d'application et de données.
* Un degré élevé de flexibilité dans la plateforme de déploiement et la configuration
* Meilleure réutilisation
* Amélioration de l'intégrité des données
* Sécurité améliorée - Le client n'a pas un accès direct à la base de données.
* Facilité de maintenance et modification aisée sans affecter les autres modules
* Dans l'architecture en trois niveaux, les performances de l'application sont bonnes.

## Méthode de modélisation

L'UML, abréviation de Unified Modeling Language (langage de modélisation unifié), est un langage de modélisation normalisé composé d'un ensemble intégré de diagrammes, développé pour aider les développeurs de systèmes et de logiciels à spécifier, visualiser, construire et documenter les artefacts des systèmes logiciels, ainsi que pour la modélisation des entreprises et d'autres systèmes non logiciels. L'UML représente un ensemble de meilleures pratiques d'ingénierie qui ont fait leurs preuves dans la modélisation de systèmes vastes et complexes. L'UML est un élément très important du développement de logiciels orientés objet et du processus de développement de logiciels. L'UML utilise principalement des notations graphiques pour exprimer la conception des projets logiciels. L'utilisation de l'UML aide les équipes de projet à communiquer, à explorer les conceptions potentielles et à valider la conception architecturale du logiciel. Dans cet article, nous vous donnerons des idées détaillées sur ce qu'est l'UML, l'histoire de l'UML et une description de chaque type de diagramme UML, ainsi que des exemples UML.[5]

UML 2 présente (14) diagrammes comme l’indiques la figure3 regroupé en deux grandes parties.

Diagrammes de structure

Les diagrammes de structure montrent la structure statique du système et de ses parties à différents niveaux d'abstraction et de mise en œuvre, ainsi que la manière dont ils sont liés les uns aux autres. Les éléments d'un diagramme de structure représentent les concepts significatifs d'un système et peuvent inclure des concepts abstraits, réels et de mise en œuvre [5] :

* Diagramme de classes
* Diagramme de composants
* Diagramme de déploiement
* Diagramme d'objets
* Diagramme de paquetage
* Diagramme de structure composite
* Diagramme de profil

Diagrammes comportementaux

Les diagrammes de comportement montrent le comportement dynamique des objets d'un système, qui peut être décrit comme une série de changements apportés au système au fil du temps. Il existe sept types de diagrammes de comportement, comme suit [8] :

* Diagramme de cas d'utilisation
* Diagramme d'activité
* Diagramme de machine à états
* Diagramme de séquence
* Diagramme de communication
* Diagramme d'aperçu des interactions
* Diagramme des temps

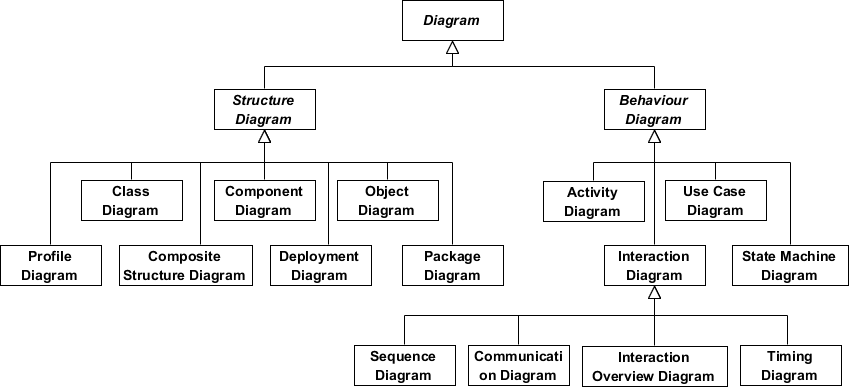


Figure 3 UML diagram structure

L'UML offre différents types de diagrammes qui nous permettent de décrire en détail les composants de notre application et leurs relations.

Dans le cadre de la conception de notre système, les trois principaux types de diagrammes utilisés sont les diagrammes de classes, les diagrammes de séquence et les diagrammes de cas d’utilisation. Voici comment nous les utilisons :

### Diagrammes de classes

* Nous utilisons les diagrammes de classes pour représenter les différentes classes de notre application. Chaque classe est représentée par un rectangle contenant son nom, ses attributs et ses méthodes.
* Les attributs représentent les caractéristiques de chaque classe, tandis que les méthodes définissent les actions qu'une classe peut effectuer.
* Les relations entre les classes sont également représentées dans les diagrammes de classes. Par exemple, nous utilisons des associations pour indiquer les liens entre les différentes classes, des agrégations pour représenter des relations tout-en-un et des généralisations pour exprimer des relations d'héritage.

### Diagrammes de séquence

* Les diagrammes de séquence nous permettent de décrire les interactions entre les différents composants de notre application lors de l'exécution de scénarios spécifiques.
* Nous utilisons ces diagrammes pour modéliser le flux des actions et les échanges de messages entre les objets. Cela nous aide à comprendre comment les différents composants interagissent les uns avec les autres et comment les données sont échangées pendant le processus de vote.
* Par exemple, nous pouvons représenter les étapes d'authentification de l'utilisateur, le chargement des données des élections, le choix d'un candidat et la validation du vote dans un diagramme de séquence.

### Diagramme de cas d'utilisation

* Le diagramme de cas d'utilisation nous permet de modéliser les différentes fonctionnalités de notre application du point de vue des acteurs qui interagissent avec elle.
* Nous identifions les acteurs, tels que les étudiants, les administrateurs ou les candidats, qui interagissent avec l'application.
* Nous représentons les cas d'utilisation qui décrivent les différentes actions que chaque acteur peut effectuer dans l'application. Par exemple, les cas d'utilisation peuvent inclure "Voter pour un candidat", "Consulter les résultats des élections" ou "Gérer les candidatures".
* Nous définissons les relations entre les acteurs et les cas d'utilisation pour indiquer quel acteur est impliqué dans chaque cas d'utilisation.

En combinant ces trois diagrammes, nous sommes en mesure de concevoir une architecture logicielle solide et bien structurée pour notre application de vote électronique. Ils nous aident à visualiser la structure, à modéliser les interactions et à clarifier les fonctionnalités de l'application, ce qui contribue à un développement efficace et à une meilleure compréhension de notre système.

## Les technologies de conception

### Framework pour le frontend

Figure 4: Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2022

#### Flutter

Flutter est un SDK assez récent (apparu en 2017) de Google. Il s'agit d'un ensemble d'outils gratuits et open-source permettant de développer des applications mobiles multiplateformes, ainsi que des solutions web, de bureau et embarquées à partir d'une base de code unique. Avec Flutter, vous pouvez développer aussi bien des MVPs que des produits complets.

Flutter est écrit en C/C++ et en Dart. Mais il utilise Dart pour le développement des applications.

Dart est aujourd'hui le langage de programmation le plus rapide pour la création d'applications iOS et Android. Il aide à créer des composants d'interface utilisateur fonctionnels, offre un excellent support IDE avec de puissantes fonctions d'auto-complétion, permet le prototypage en direct sans perdre l'état de l'application, et nécessite une programmation orientée objet. Il vous permet de créer des interfaces utilisateur flexibles et expressives qui fonctionnent en mode natif.

Flutter se compose de deux parties :

Le SDK (Software Development Kit), qui est un ensemble d'outils pour le développement, le débogage, le profilage, etc.

Le framework lui-même, qui est une bibliothèque d'interface utilisateur basée sur des widgets. Elle est représentée par une collection de boutons, de textes, de curseurs, etc. que vous pouvez personnaliser et créer votre propre design.

Les applications basées sur Flutter sont construites à partir d'une base de code unique, et le langage de programmation Dart est compilé directement en code natif, ce qui augmente les performances de ces applications. Elles utilisent une unité de traitement graphique (GPU) pour le rendu et peuvent accéder aux API et aux services de la plateforme (appareil photo, stockage local, etc.).[6]

#### Avantages et inconvénients

Tableau 1 Flutter Avantages et inconvénients

|  |  |  |
| --- | --- | --- |
| **Framework** | **Pour** | **Cons** |
|  | * Développement rapide * Rechargement à chaud * Des performances rapides * Un riche ensemble de widgets personnalisables * Intégrations * Une interface utilisateur flexible et expressive * Applications natives * Open source * Facile à apprendre * Une excellente documentation * Flutter est idéal pour les débutants | * Flutter est une technologie jeune, donc il peut y avoir des bugs, des problèmes liés aux mises à jour, et toutes les autres "beautés" d'un jeune outil. * Dart n'est pas un langage populaire. * Un PC puissant est nécessaire. |

### Choix de notre framework

Pour le développement de notre application mobile, nous avons pris la décision d'utiliser Flutter comme framework. Flutter, développé par Google, est un framework open-source et multiplateforme qui permet la création d'applications natives pour iOS et Android à partir d'un seul codebase commun. Depuis 2019, il est forte croissance en termes d’utilisation selon la figure 4.

## Backend

### Firebase

Pour la partie backend de notre application, nous avons choisi d'utiliser Firebase. Firebase est une plateforme de développement d'applications mobiles et web proposée par Google, offrant un ensemble d'outils et de services cloud puissants pour la création et la gestion d'applications.

### Avantages & Limites de Firebase

Tableau 2 Avantages & Limites de firebase

|  |  |  |
| --- | --- | --- |
|  | **Avantages** | **Limites** |
|  | * Démarrage gratuit * Vitesse de développement * Plate-forme de développement d’applications de bout en bout * Proposé par Google * Les développeurs peuvent se concentrer sur le développement frontend * Il n’y a pas de serveur * Il offre des capacités d’apprentissage automatique * Génère du trafic vers vos applications * Contrôle des erreurs * Sécurité | * Il n’est pas Open-Souce * Verrouillage du fournisseur * Firebase ne fonctionne pas dans de nombreux pays * Seules les bases de données NoSQL sont disponibles * Requêtes lentes * Tous les services ne sont pas gratuits au départ * Fonctionne uniquement sur Google Cloud * Les serveurs dédiés et le soutien aux entreprises ne sont pas une option * Ne fournit pas d’API GraphQL |

# Biliographie

[1] Issoufou, Sanov. « Le phénomène tontinier au Burkina Faso : étude sur 69 cas ». *Revue internationale P.M.E. Économie et gestion de la petite et moyenne entreprise* 5, no 3‑4 (1992): 153‑70. https://doi.org/10.7202/1008159ar.

[2] Tontine Trust. « About Tontine Trust ». Consulté le 6 juillet 2023. <https://tontine.com/about-us/>.

[3]« Tontine ».In*Wikipédia*,7juin2023. https://fr.wikipedia.org/w/index.php?title=Tontine&oldid=204989569#cite\_note-8.

[4] support@baianat.com, “Design for users,” *Baianat*. https://www.baianat.com/books/ui-as-an-asset-for-ux/design-for-users (accessed Jun. 15, 2023).

[5] “What is Unified Modeling Language (UML)?” https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/;WWWSESSIONID=B90B026F444CD918FE9CA2896670D50B.www1 (accessed Jul. 01, 2023).

[9] “Flutter Which is Better for 2023? | SaM Solutions,” Mar. 14, 2023. https://www.sam-solutions.com:443/blog/flutter-vs-react-native/ (accessed Jun. 29, 2023).