

## Table des matières

Introduction	3
Contexte	3
Introduction	3
Le client	3
La demande initiale	4
Etude de marché	4
Les produits existants	5
Les produits existant pour iOS	6
Analyse des besoins	7
Fonctionnalités	7
Fonctionnalités non retenues	8
Utilisateurs	8
Interface utilisateur	9
Sécurité	11
Plateformes	11
Mobile	11
Serveur	12
Conception	12
Cahier des charges	12
Partie visuelle	14
Diagrammes	15
Diagramme de cas d'utilisation	15
Diagramme de séquence – ouverture de l'application	16
Diagramme de séquence – hub	17
Diagramme de séquence – chronomètre	18
Diagramme de séquence – synchronisation	19
Développement	20
Dépôt du projet et versioning	20
Structure du projet	21
Langages « cross-platform »	22
Langage Swift (Apple)	23
Communications entre iOS et Odoo	24
L'identification d'un utilisateur	25
Lecture d'une table de la base de données	26

Liste des méthodes possible pour le protocole JSON-RPC avec Odoo	27
Réalisation du code informatique	28
L'orienté objet	28
Réalisation des objets	28
L'objet User	29
L'objet Settings	29
L'objet Project	30
L'objet Task	30
L'objet Record	31
Sauvegarde des données internes	31
Requêtes HTTPS	32
Code de l'interface utilisateur (SwiftUI)	33
Test et validation	35
Environnement de test Odoo et Postman	35
Environnement de test xCode	36
Validation auprès de Mr Burniaux	36
Futur de l'application	37
Conclusion	37
Récapitulation des objectifs	37
Résumé des résultats	38
Contribution à la connaissance	38
Limitations et recommandations futures	39
Conclusion générale	39
Remerciements	40
Webographie et documentations	41
Annexes	41

## Introduction

Dans le cadre de ma dernière année de bachelier en informatique à l'école IETC-ps de Charleroi, j'ai eu l'occasion de travailler sur un projet de fin d'études pour la société Burniaux-consulting.

Ce travail représente l'occasion pour moi de mettre en pratique les compétences et les connaissances acquises tout au long de ma formation en informatique. Mon objectif est de concevoir une application répondant à un besoin spécifique en utilisant une méthodologie de développement et des outils et technologies modernes.

Je décide de concevoir cette application qui aura pour but de résoudre un problème concret et d'améliorer la vie quotidienne des utilisateurs finaux. Afin de réaliser cet objectif, je vais m'efforcer d'utiliser les technologies les plus appropriées et de suivre les meilleures pratiques de développement pour garantir la qualité et les performances de l'application.

## Contexte

### Introduction

Aujourd'hui, les entreprises de services professionnels cherchent constamment à maximiser leur efficacité et leur rentabilité. Dans ce contexte, le suivi précis du temps de travail consacré à chaque projet et tâche spécifique est essentiel. Cependant, la collecte manuelle de ces données peut être fastidieuse et sujette à des erreurs. En intégrant un moyen rapide et ergonomique de se connecter à Odoo, non seulement cela permettrait aux entreprises de gagner du temps, mais cela faciliterait également la saisie précise des informations de temps de travail.

C'est dans ce contexte que l'idée de développer une application mobile pour enregistrer les valeurs de temps liées aux tâches est apparue. Cette application permettrait aux clients de Burniaux consulting de facilement enregistrer et suivre leur temps de travail sur différents projets, en utilisant leur téléphone portable ou leur tablette.

Ce travail de fin d'étude vise donc à concevoir et développer une application mobile conviviale et fonctionnelle, qui répond aux besoins spécifiques de Burniaux consulting. L'objectif final est de fournir à la société un outil de suivi de temps efficace et précis, permettant de facturer les projets en toute transparence et de maximiser leur rentabilité.

### Le client

Monsieur François Burniaux est le fondateur et propriétaire de la société de consultance en informatique "Burniaux Consulting". Cette société se spécialise dans l'intégration du logiciel Odoo chez ses clients, ainsi que dans la fourniture d'autres services de consultance informatique. En plus de son rôle de chef d'entreprise, Monsieur Burniaux est également professeur à l'IETC-ps, où il enseigne les cours de "POO", "Structure des ordinateurs", "système d'exploitation" et "ERP". C'est grâce à son cours "ERP" que j'ai connu Monsieur Burniaux. En tant que client pour mon travail de fin d'études, Monsieur Burniaux représente un professionnel de l'informatique avec une expertise dans les logiciels ERP, en particulier dans Odoo.

## La demande initiale

Le client souhaite la conception d'une application mobile de gestion de temps comprenant un chronomètre. Cette application doit permettre aux utilisateurs de s'identifier avant de l'utiliser, et doit être capable de fonctionner hors ligne pour le comptage de temps. En outre, elle doit être en mesure de se connecter à l'erp Odoo pour échanger des données, notamment l'identification des utilisateurs, la récupération de la liste des projets et des tâches qui y sont liées, ainsi que l'inscription de valeurs de temps chronométrées dans la base de données d'Odoo pour une analyse ou une facturation ultérieure.

Par "application mobile", il est entendu que l'application doit être native et installée sur des appareils mobiles tels que des smartphones.



## Etude de marché

Avant de commencer le développement de l'application, il est important de mener une étude de marché pour examiner les produits existants sur le marché et pour identifier les lacunes à combler. Dans cette section, nous allons examiner les différentes applications existantes de comptage de temps qui existent sur le marché. Nous allons également discuter de leurs fonctionnalités et de leurs avantages, ainsi que de leurs limites.

## Les produits existants

**Toggl** : Toggl est une application de gestion de temps populaire qui permet aux utilisateurs de suivre leur temps de travail en temps réel, de définir des limites de temps et de facturer leurs clients en conséquence. Elle offre une interface facile à utiliser et une intégration avec de nombreuses autres applications.

<https://toggl.com/track/>

**RescueTime** : RescueTime est une application de gestion de temps qui suit le temps passé sur différentes applications et sites Web, et fournit des rapports détaillés sur la productivité des utilisateurs. Elle offre également des fonctionnalités de blocage de site pour aider les utilisateurs à se concentrer.

<https://www.rescuetime.com/>

**Clockify** : Clockify est une application de suivi du temps qui permet aux utilisateurs de suivre leur temps de travail sur des projets spécifiques et de générer des rapports pour la facturation et l'analyse. Elle offre une version gratuite avec des fonctionnalités de base, ainsi que des options payantes pour des fonctionnalités avancées.

<https://clockify.me/fr/>

**Trello** : Trello est une application de gestion de projet qui permet aux utilisateurs de créer des tableaux pour suivre les tâches et les projets en cours. Elle offre également des fonctionnalités de suivi du temps pour aider les utilisateurs à estimer le temps nécessaire pour chaque tâche et à suivre leur temps de travail.

<https://trello.com/home>

**Harvest** : Harvest est une application de gestion de temps et de facturation qui permet aux utilisateurs de suivre leur temps de travail, de créer des factures pour leurs clients et de gérer leur budget de projet. Elle offre également des options de suivi de projet pour aider les utilisateurs à rester organisés.

<https://www.getharvest.com>

Ces produits offrent des fonctionnalités de base similaires en matière de suivi du temps, de facturation et de gestion de projet, mais diffèrent en termes de convivialité, d'intégrations avec d'autres applications et de coûts.

Enfin, on peut remarquer que la plupart des produits existants sur le marché de la gestion de temps sont des services en ligne accessibles via un navigateur Web. Ils offrent des fonctionnalités étendues pour la gestion de projet, de facturation et de suivi du temps, mais ne sont pas spécifiquement conçus pour l'interaction avec un serveur Odoo. De plus, la plupart ne sont pas natifs sur application mobile et peuvent offrir un ensemble de fonctionnalités trop avancé pour les besoins spécifiques d'une application mobile de gestion de temps avec un chronomètre. Ainsi, il est nécessaire de chercher des solutions adaptées pour répondre aux besoins particuliers de chaque entreprise ou projet.

## Les produits existant pour iOS

**HoursTracker** (Gratuit avec des achats intégrés) - Une application de suivi du temps qui vous permet de créer des tâches, de définir des tarifs et de générer des rapports.

<https://www.hourstrackercloud.com/>

**Timely** (Gratuit avec des achats intégrés) - Cette application offre une approche automatique du suivi du temps. Elle enregistre automatiquement le temps passé sur différentes activités et génère des rapports détaillés.

<https://timelyapp.com/home-a>

**Tyme 2** (Payant) - Une application de gestion du temps qui vous permet de suivre les heures facturables, de définir des budgets et de visualiser les statistiques sous forme de graphiques.

<https://apps.apple.com/fr/app/tyme-2/id1063996724?mt=12>

**ATracker** (Gratuit avec des achats intégrés) - Cette application vous permet de suivre le temps passé sur des tâches spécifiques à l'aide d'une interface simple. Vous pouvez créer des catégories personnalisées et générer des rapports détaillés.

<https://atracker.pro/home.html>

**Hours** (Gratuit avec des achats intégrés) - Une application de suivi du temps intuitive qui vous permet de définir des projets, de suivre le temps passé et de générer des rapports visuels.

<https://apps.apple.com/us/app/hours-time-tracking/id895933956>

**Timing** (Payant) - Une application de suivi du temps qui enregistre automatiquement le temps passé sur différentes applications et sites Web. Elle vous offre également des fonctionnalités de facturation et de rapports détaillés.

<https://timingapp.com/?lang=fr>

**Toggl Track** (Gratuit avec des achats intégrés) - Une version mobile de Toggl, cette application vous permet de suivre le temps passé sur des tâches, de synchroniser vos données et de générer des rapports.

<https://toggl.com/>

**Timeneye** (Gratuit avec des achats intégrés) - Cette application vous permet de suivre le temps passé sur des projets et des tâches spécifiques. Elle offre également des fonctionnalités de planification et de collaboration.

<https://www.timeneye.com/>

Tout comme la liste des produits mentionnée précédemment, ces produits ne répondent pas aux besoins spécifiques d'une application mobile capable d'interagir avec l'ERP Odoo, même s'ils sont nativement installables sur iOS. La plupart de ces produits utilisent leur propre format de données et ne proposent pas de connecteur spécifique à Odoo. Il convient de noter que certains d'entre eux sont payants.

## Analyse des besoins

L'analyse des besoins permet de bien comprendre les exigences du client et de ses utilisateurs, et de les traduire en exigences fonctionnelles et non-fonctionnelles.

### Fonctionnalités

L'objectif est de créer un chronomètre intelligent qui peut se connecter à un serveur distant Odoo pour enregistrer les temps effectués sur les tâches.

Voici donc les fonctionnalités principales que l'application doit posséder :

1. **Connexion à un serveur distant Odoo** : L'application doit permettre à l'utilisateur de se connecter à un serveur distant Odoo en renseignant l'URL de connexion, le nom de la base de données, ainsi que son login et son mot de passe.
2. **Récupération des listes de projets et tâches** : L'application doit être capable de récupérer la liste des projets ainsi que les tâches qui y sont liées à partir du serveur Odoo. Ces informations doivent être sauvegardées dans la mémoire de l'appareil pour permettre une utilisation hors-ligne de l'application.
3. **Interface de chronométrage** : L'application doit proposer une interface de chronométrage complète qui permettra à l'utilisateur de démarrer, mettre en pause, et arrêter le chronomètre. Cette interface devra également afficher les temps enregistrés.
4. **Interface de chronométrage manuelle** : L'application doit également proposer une interface alternative qui permettra à l'utilisateur de spécifier des données de temps de manière manuelle.
5. **Enregistrement des temps effectués sur les tâches** : L'application doit pouvoir enregistrer les temps effectués sur les tâches dans la mémoire de l'appareil de manière sécurisée, afin que l'utilisateur puisse y accéder ultérieurement. L'application doit également permettre la synchronisation des temps avec le serveur distant Odoo lorsque l'utilisateur a accès à une connexion Internet, ou de les supprimer.
6. **Affichage des temps enregistrés** : L'application doit pouvoir afficher les temps enregistrés par l'utilisateur et proposer une action qui permette à l'utilisateur de supprimer un temps enregistré.
7. **Synchronisation des temps avec le serveur** : Lorsque l'utilisateur le souhaite, l'application doit être en mesure de synchroniser les temps enregistrés sur le serveur Odoo. L'application doit pouvoir enregistrer les temps dans sa mémoire interne même en l'absence de connexion Internet.

En résumé, l'application mobile doit être en mesure de permettre à l'utilisateur de se connecter à un serveur Odoo, d'enregistrer les temps effectués sur les tâches dans la mémoire de l'appareil, de synchroniser les temps avec le serveur Odoo, d'afficher les temps enregistrés, et de récupérer la liste des projets et tâches à partir du serveur distant.

## Fonctionnalités non retenues

Lors de la création de cette application, certaines fonctionnalités sont écartées pour diverses raisons expliquées ici :

1. **Création de projet et de tâches** : L'application ne permet pas aux utilisateurs de créer de nouveaux projets ou de définir des tâches spécifiques. Son objectif principal est de faciliter la connexion et l'accès aux fonctionnalités existantes sur le serveur Odoo.
2. **Consultation de profil sur le serveur Odoo** : L'application ne fournit pas la possibilité de consulter en détail les profils des utilisateurs sur le serveur Odoo. Elle se concentre uniquement sur l'authentification et la connexion des utilisateurs.
3. **Création automatique de facture** : L'application ne propose pas la fonctionnalité de créer automatiquement des factures. Les utilisateurs devront effectuer cette tâche spécifique via l'ERP Odoo, qui offre des fonctionnalités complètes de gestion des factures.
4. **Consultation des données déjà enregistrées sur le serveur Odoo** : L'application ne permet pas la consultation des données précédemment enregistrées sur le serveur Odoo. Son objectif principal est de faciliter la connexion et l'accès aux fonctionnalités en temps réel sans stocker localement les données déjà présentes sur le serveur Odoo.

## Utilisateurs

Les clients de la société Burniaux Consulting, qui utilisent le logiciel Odoo, constituent une part importante des utilisateurs. Ils bénéficient de l'application interne d'Odoo nommée "project" pour la gestion de leurs projets. Cependant, l'idée de cette application est de proposer une expérience plus conviviale, offrant une approche alternative pour travailler avec Odoo, tout en profitant des avantages offerts par un smartphone par rapport à un PC.

Plus précisément, ces clients souhaitent pouvoir comptabiliser facilement le temps passé sur chaque tâche du projet. On peut voir en cette fonctionnalité une opportunité de planifier plus efficacement leur travail, d'évaluer les coûts des projets, de faciliter la facturation et d'améliorer leur productivité.

Ainsi, l'analyse des besoins des utilisateurs met en évidence d'intégrer une fonctionnalité de gestion du temps pour les tâches des projets dans l'application "project" d'Odoo. Cette fonctionnalité offrirait aux clients de la société Burniaux Consulting une manière plus agréable de travailler avec Odoo, tout en leur permettant de mieux gérer leurs ressources et leur temps, et d'accroître leur efficacité et leur rentabilité.



## Interface utilisateur

L'interface utilisateur, également appelée interface graphique ou interface homme-machine, est la partie de tout système informatique qui permet à l'utilisateur de communiquer avec celui-ci. Elle est essentielle pour rendre l'utilisation d'un logiciel ou d'un système informatique plus intuitive et plus conviviale. L'interface utilisateur peut être constituée de divers éléments, tels que des boutons, des menus, des fenêtres, des icônes, des curseurs, des boîtes de dialogue, etc.

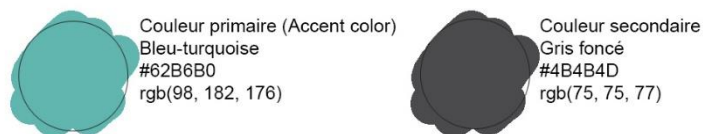
L'objectif principal d'une interface utilisateur est de permettre à l'utilisateur de réaliser facilement et efficacement les tâches qu'il souhaite effectuer avec l'application. Une interface utilisateur bien conçue doit être simple, intuitive et facile à utiliser, afin que l'utilisateur n'ait pas besoin de passer du temps à apprendre à utiliser le système. Elle doit également fournir un retour d'information clair et précis sur les actions effectuées par l'utilisateur et sur l'état du système par exemple sur un retour d'erreur.

Dans le cadre de mon travail de fin d'étude, j'ai reçu une charte graphique de la part de M. Burniaux. Cette charte comprenait un thème général, un logo d'entreprise et une suite de couleurs à utiliser. L'objectif de cette charte graphique est de garantir une cohérence visuelle dans l'ensemble de mon travail, en utilisant les couleurs et les éléments graphiques préétablis. En respectant cette charte graphique, on peut s'assurer que le travail a une identité visuelle professionnelle et cohérente.

Voici une première ébauche de l'interface utilisateur de l'application. Cette ébauche permet de mieux comprendre comment l'application fonctionnera et donne une direction claire pour la conception future.



On peut remarquer que les couleurs prédominantes du logo de l'entreprise sont celles-ci



*A la demande de Mr Burniaux, il est demandé de n'utiliser que la couleur primaire concernant les boutons. On peut remarquer que dans l'ébauche le bouton de connexion est un dégradé de bleu/violet qu'il faudra remplacer par la couleur primaire.*

## Sécurité

La sécurité est un aspect essentiel pour toute application, notamment pour garantir la confidentialité et l'intégrité des données utilisateur. Voici quelques éléments essentiels à inclure dans l'analyse des besoins en matière de sécurité :

1. **Utilisation du protocole HTTPS** : L'utilisation de HTTPS est essentielle pour protéger les données transitant entre l'application mobile et le serveur. Le protocole HTTPS utilise un cryptage SSL/TLS pour chiffrer les données, empêchant ainsi toute interception ou altération des données en transit.
2. **Vérification de l'identité** : On peut utiliser le serveur Odoo pour vérifier l'identité de l'utilisateur lorsqu'il se connecte à l'application. On peut également utiliser des mécanismes d'authentification tels que la vérification par clé API pour renforcer la sécurité de l'application. Une option existe sur Odoo qui permet à un utilisateur de générer une clé unique d'identification, cette clé unique ne peut servir que pour un seul utilisateur et sur une seule application tierce. (Clé api non implémentée dans l'application car non demandé).
3. **Gestion des fichiers** : iOS fournit un environnement sécurisé pour les applications, ce qui signifie que les fichiers créés par une application ne sont accessibles que par cette dernière. On peut utiliser cette fonctionnalité pour garantir que les données de l'application ne sont pas accessibles à d'autres applications ou à des utilisateurs non autorisés.
4. **Protection contre les attaques de force brute** : Les attaques de force brute sont des tentatives répétées pour deviner un mot de passe ou une clé d'accès. On peut inclure des mécanismes de protection tels que la mise en place de limites de connexion. (Non implémenté dans l'application car non demandé).

Il est important de s'assurer que l'application offre un niveau de sécurité suffisant pour protéger les données des utilisateurs et garantir la confidentialité et l'intégrité des données.

## Plateformes

### Mobile

Dans le cadre de ce projet, le terme "plateforme mobile" fait référence à un appareil qui diffère d'un ordinateur classique. Au cours de ma formation, l'accent a toujours été mis sur la programmation sur ordinateur, ce qui me laisse dans l'incertitude au premier abord, bien que la programmation puisse être similaire.

Un appareil mobile peut prendre la forme d'un smartphone, c'est-à-dire un téléphone portable capable d'installer des applications tierces et de se connecter à Internet. Il peut également s'agir d'une tablette. Le terme "plateforme mobile" englobe un large éventail d'appareils.

Après avoir discuté avec M. Burniaux, il a été convenu de se concentrer exclusivement sur les appareils mobiles de la marque Apple, tels que l'iPhone et l'iPad. Cette décision a été prise car les langages "cross-platform" disponibles ne satisfaisaient pas les exigences de ce projet en termes de performance (voir plus loin). J'ai donc dû opter pour un langage propriétaire.

Les deux choix réalistes étaient donc Apple et Android, qui sont les systèmes d'exploitation mobile les plus répandus. M. Burniaux a choisi Apple, car cela devrait correspondre à la majorité de sa clientèle.

## Serveur

Un serveur qui tourne avec un logiciel ERP comme Odoo est un système informatique dédié qui héberge et exécute le logiciel Odoo pour gérer les processus et les données d'une entreprise. Il agit comme une plateforme centrale où les informations sont stockées, traitées et accessibles aux utilisateurs autorisés via des dispositifs connectés. Le serveur assure le bon fonctionnement du logiciel ERP et facilite la gestion des différentes activités commerciales telles que la comptabilité, les ressources humaines, les ventes, les achats, etc.

Burniaux Consulting est une société spécialisée dans l'installation et la configuration de logiciels ERP tels qu'Odoo pour ses clients. Pour le développement de ce projet, M. Burniaux mis à ma disposition un serveur de test à l'adresse [demo.burniaux.com](http://demo.burniaux.com). Cela permet de réaliser l'application dans un environnement dédié et de procéder aux tests nécessaires.

Odoo est un logiciel ERP (Enterprise Resource Planning) open source, gratuit et complet. Il fournit une suite d'applications intégrées qui couvrent différents aspects de la gestion d'une entreprise, tels que la comptabilité, la gestion des ressources humaines, la gestion des ventes, des achats, de la fabrication, du marketing, de la gestion de projet, du CRM (Customer Relationship Management) et bien plus encore.

Odoo est conçu pour aider les entreprises à centraliser et à automatiser leurs processus opérationnels, à améliorer leur efficacité et leur productivité, et à faciliter la collaboration entre les différents départements et équipes. Il permet de gérer et d'analyser les données de l'entreprise de manière intégrée, offrant ainsi une vision globale et précise de ses activités.

L'un des avantages d'Odoo réside dans sa flexibilité et sa capacité d'adaptation aux besoins spécifiques de chaque entreprise. Il permet la personnalisation des flux de travail, des tableaux de bord, des rapports et des fonctionnalités selon les exigences de l'entreprise.

Odoo est également soutenu par une large communauté d'utilisateurs et de développeurs, ce qui assure un support continu, des mises à jour régulières et une évolutivité du système.

## Conception

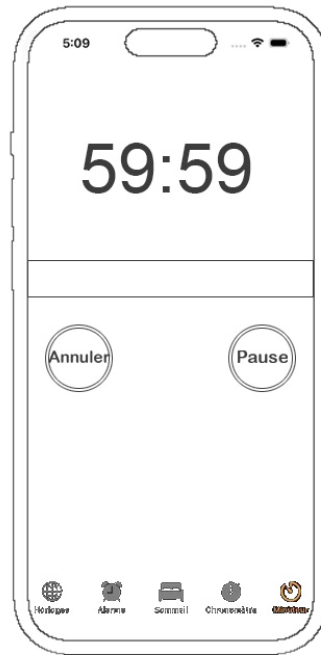
### Cahier des charges

Le cahier des charges est un document important pour tout projet informatique. Son rôle est de définir les spécifications et les exigences du projet, ainsi que de décrire les fonctionnalités, les exigences techniques, les délais, les coûts et les contraintes de mise en œuvre. Dans le cadre de ce travail de fin d'étude en Informatique de gestion, ce document sera utilisé comme un guide pour décrire de manière précise et détaillée les besoins du client en matière d'application.

Cahier des charges				
Réalisation d'une application de chronomètre	MVP	Option	Réalisé	
Application				
L'application doit être native, c'est-à-dire être compilée et installée localement sur un périphérique mobile	✓			✓
L'application doit correspondre à une charte graphique donnée et y inclure des logo de l'entreprise "Burniaux consulting"	✓			✓
Connexion				
L'application doit pouvoir se connecter à un serveur distant odoo	✓			✓
L'application doit permettre une authentification de l'utilisateur (vérification login/mdp)	✓			✓
Si possible, permettre une authentification par autre méthode (FaceID, empreinte digitale, ...)		✓		✓
L'application doit également permettre de se déconnecter du serveur lorsque l'utilisateur le souhaite	✓			✓
Fonctionnalités				
L'application doit proposer une interface de chronométrage classique (bouton play, stop, pause, ...)	✓			✓
Permettre de sauvegarder les temps enregistrés sur des taches définies	✓			✓
Permettre une consultation des données enregistrées en attente de synchronisation sur le serveur odoo	✓			✓
Lors de la consultation des données, permettre à l'utilisateur de les supprimer	✓			✓
L'application pourrait également proposer une interface de chronométrage manuelle, afin de rentrer des données brutes		✓		✓
Odoo				
L'application doit permettre une authentification de l'utilisateur grâce au serveur odoo	✓			✓
L'application doit pouvoir se synchroniser avec le serveur odoo, et récupérer la liste de projets et taches qui y sont définis	✓			✓
Dans le processus de synchronisation l'application doit pouvoir enregistrer ses données de temps dans la base de données d'odoo	✓			✓
Une fois que l'application s'est synchronisée celle-ci doit effacer les données de temps de sa mémoire interne si la synchronisation est terminée (processus asynchrone)	✓			✓
Sécurité				
Le protocole à utiliser pour le transfert de données vers le serveur est en HTTPS	✓			✓
Les données reçues du serveur doivent être enregistrées de manières sécurisées sur le mobile	✓			✓
Les données enregistrées grâce à l'interface de chronométrage doivent être sécurisée également	✓			✓

## Partie visuelle

En m'appuyant sur la charte graphique fournie, j'ai trouvé une source d'inspiration dans les applications natives déjà existantes, telles que l'application de chronomètre intégrée à iOS (étant donné que l'application devra être initialement proposée pour le système Apple). J'ai pu retrouver des éléments de conception de base, tels que les boutons Play, Stop, etc., ainsi qu'un menu situé en bas de l'écran pour permettre la navigation entre les différentes fonctionnalités attendues.

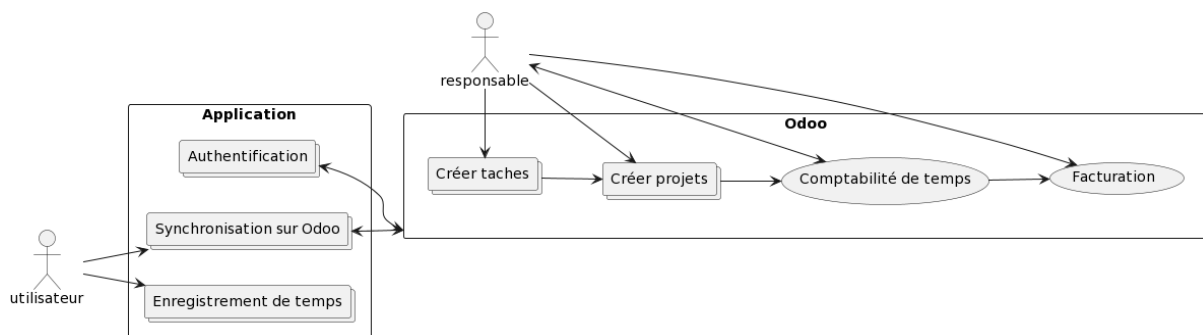


*Exemple de l'application de chronomètre intégrée à iOS.*

## Diagrammes

### Diagramme de cas d'utilisation

Ce schéma de cas d'utilisation présente de manière globale le fonctionnement de l'application et son utilisation en relation avec Odoo. Tout d'abord, le responsable doit créer un projet et y attribuer des tâches. Ces projets et tâches sont ensuite récupérés par l'application, qui propose à l'utilisateur un menu pour enregistrer les données de temps sur une tâche spécifique. Une fois ces données enregistrées, l'utilisateur doit lancer une synchronisation avec le serveur Odoo pour que les données soient prises en compte dans le système. Ensuite, il incombe au responsable de gérer la comptabilité du temps passé sur les projets et de générer une facture correspondante.

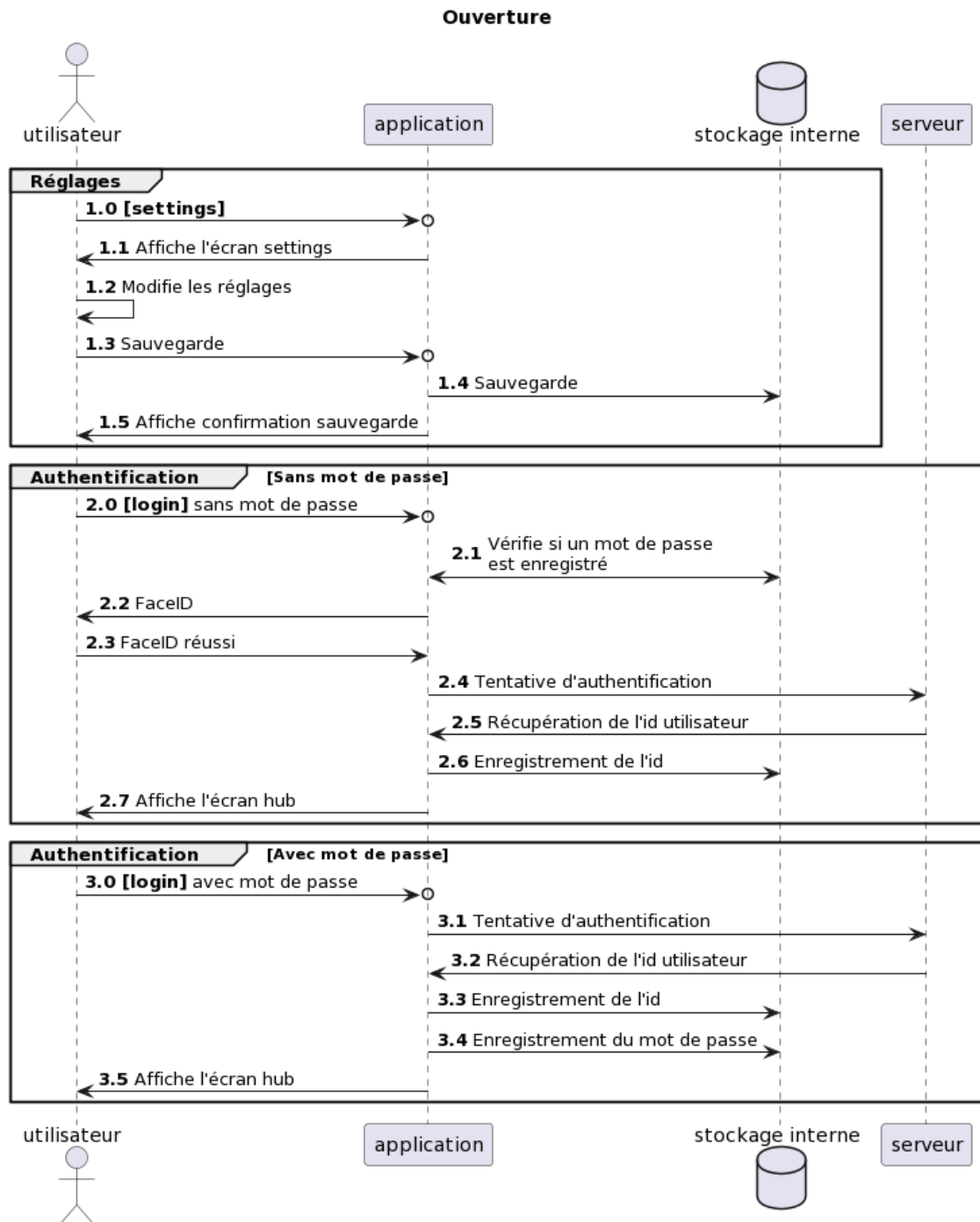


Comme mentionné dans une partie précédente l'application n'est pas intégrée directement dans le serveur Odoo, mais elle interagit avec celui-ci. Elle est conçue pour être autonome et doit être capable de fonctionner sans connexion internet, sauf lors de la synchronisation. L'application permet uniquement d'insérer des données de temps sur des tâches. Conformément à la demande de M. Burniaux, il n'est pas possible de créer des projets et des tâches à partir de l'application elle-même ; cela doit être effectué à l'aide d'un module sur Odoo.

Plusieurs scénarios peuvent être envisagés. Un premier concerne un employé de la société de M. Burniaux qui enregistre le temps passé lors de la création de programmes informatiques. M. Burniaux peut ainsi vérifier le temps passé sur les projets en cours, créant ainsi une donnée analytique supplémentaire. Dans un deuxième scénario, on peut imaginer un prestataire de services qui chronomètre ses prestations et enregistre ces données pour une facturation automatique. Dans ce cas, l'utilisateur et le responsable seraient la même personne.

## Diagramme de séquence – ouverture de l'application

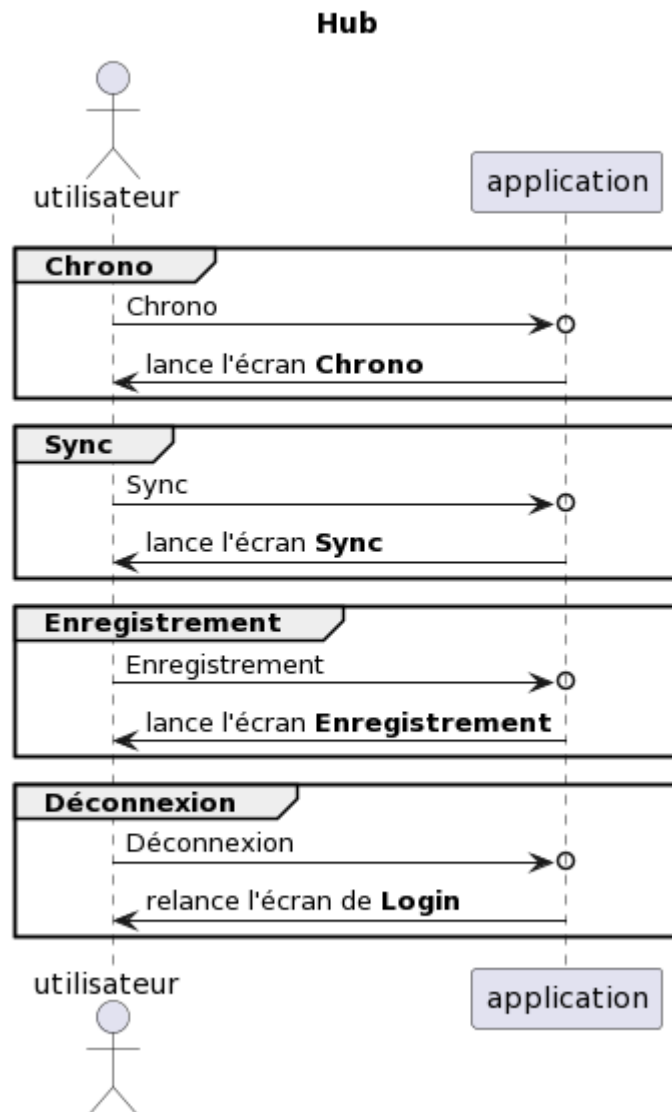
Le diagramme de séquence présenté représente la première interface de l'application. Celle-ci s'affiche dès que l'utilisateur lance l'application, et permet d'accéder à plusieurs fonctionnalités. Le diagramme détaille également les réactions de l'application en réponse aux actions de l'utilisateur. On observe notamment que l'utilisateur peut accéder à un deuxième écran pour les réglages, ainsi qu'à une fonctionnalité d'authentification avec ou sans mot de passe, qui témoigne de la liaison de l'application avec le serveur Odoo.





### Diagramme de séquence – hub

Le diagramme illustre le hub de l'application, qui est la partie centrale accessible après identification. Le hub joue un rôle essentiel en gérant les boutons de menu de l'application, situés dans la partie inférieure de l'écran conformément à la philosophie des applications Apple. C'est à travers ce code que les boutons de menu sont gérés, offrant ainsi à l'utilisateur un point d'entrée principal pour accéder aux différentes fonctionnalités de l'application.

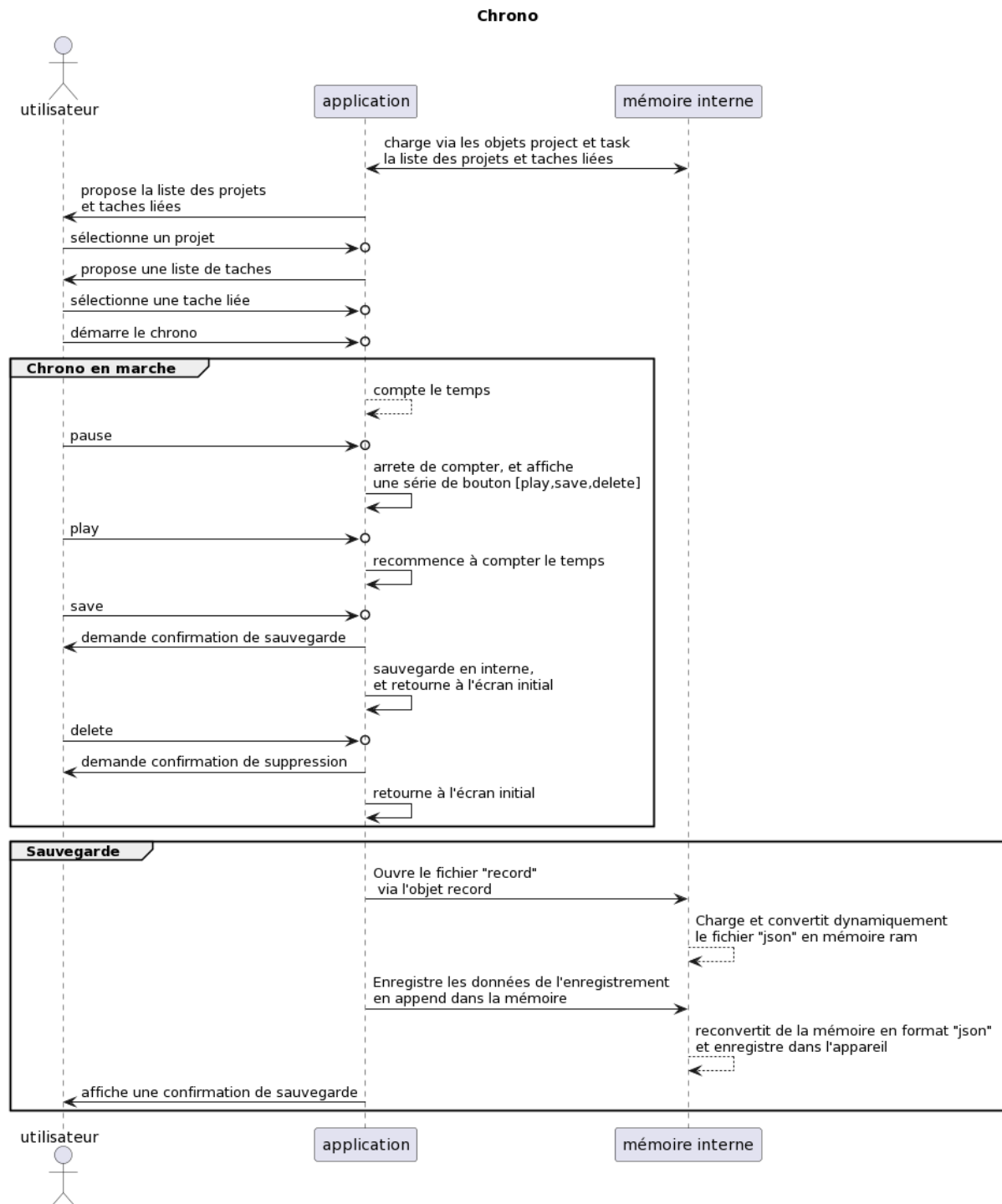


Dans le contexte d'une application, un hub est un point central qui permet de regrouper et de gérer plusieurs fonctionnalités, services ou appareils connexes. Il peut être considéré comme un nœud central qui facilite la communication, la coordination et l'intégration entre différents éléments de l'application.

## Diagramme de séquence – chronomètre

Ce diagramme illustre les parties fonctionnelles du chronomètre. On peut remarquer que l'utilisateur doit toujours au préalable choisir un projet parmi une liste proposée. Une fois fait l'application affiche une deuxième liste proposant des tâches qui sont liées au projet choisi.

Une fois lancé l'application affiche un ensemble de fonctionnalité standards que l'on retrouve dans les applications similaires dont l'ordre de fonctionnement est détaillé dans ce diagramme.

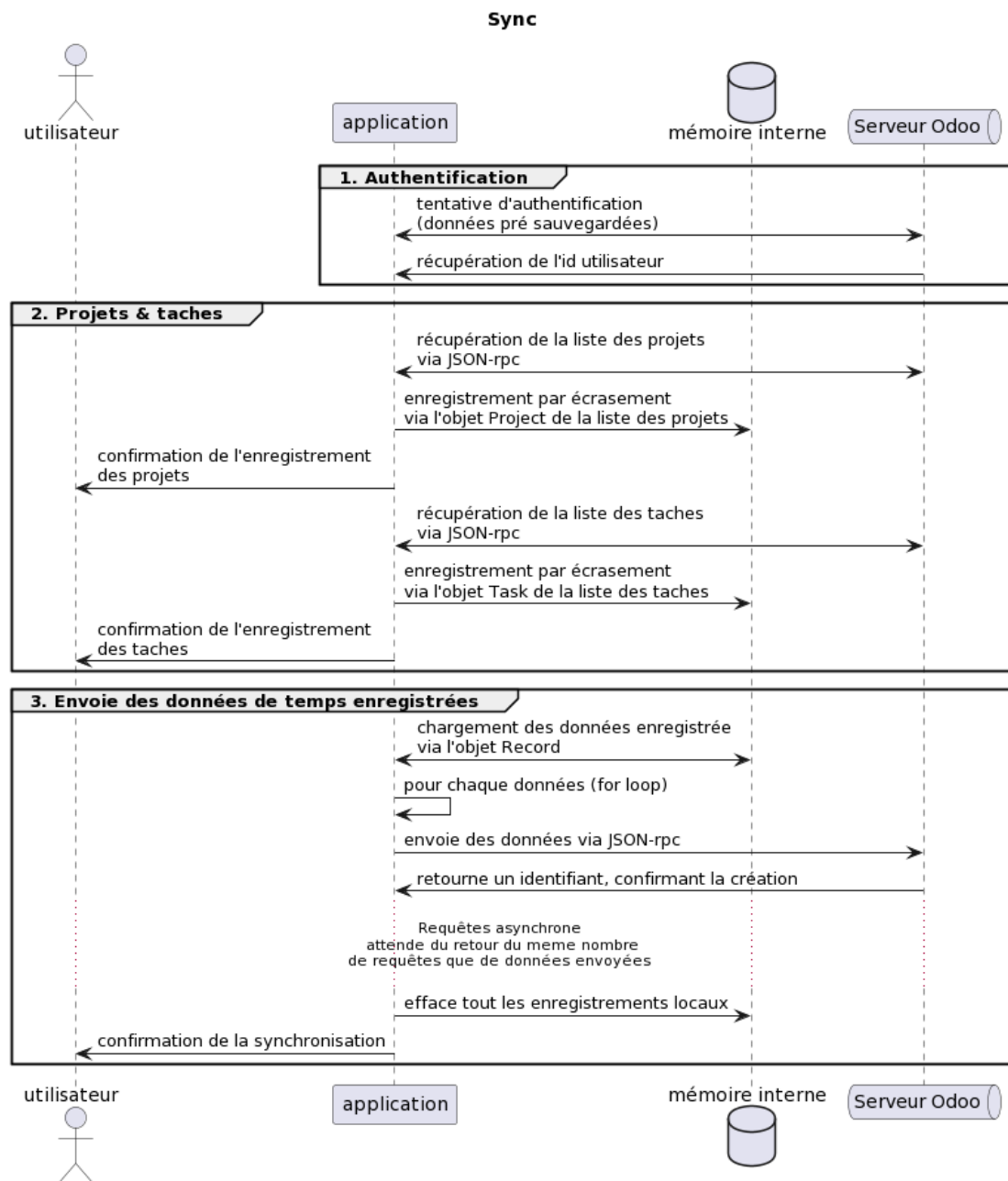


## Diagramme de séquence – synchronisation

Ce diagramme aide à comprendre le processus de synchronisation entre l'application et un serveur Odoo. Il est important de noter que c'est grâce à ce processus que l'application récupère la liste des projets et des tâches, il est donc nécessaire de lancer ce processus au minimum une fois avant de pouvoir lancer un chronomètre lorsque l'application n'a jamais été utilisée.

Le processus de synchronisation refait une vérification d'authentification par sécurité vers le serveur Odoo pour des raisons de sécurité.

Une fois que le processus est lancé, l'application enverra données par données sous forme de requête Https et en utilisant le protocole JSON-RPC au serveur Odoo. Comme ces requêtes sont asynchrone, l'application compte combien de données elle envoie au serveur et combien de confirmation elle reçoit. Le processus ne se termine que lorsque le nombre de requêtes envoyées est égales aux nombres de réponses de la part du serveur. Une fois terminé, l'application supprime toutes les données enregistrées concernant les temps.



## Développement

### Dépôt du projet et versioning

Afin de rendre le code source de l'application disponible et consultable par un ensemble de personnes le souhaitant, j'ai décidé d'utiliser le service en ligne GitHub.

GitHub est un service web, un système de gestion de versions distribué. Il permet aux développeurs de stocker, gérer et partager leur code source ainsi que de collaborer avec d'autres développeurs à travers le monde. GitHub permet également de suivre les modifications apportées à un projet et de revenir en arrière en cas de besoin.

Voici quelques-uns des avantages de l'utilisation de GitHub pour la gestion de versions de code :

- Versioning : GitHub permet de conserver l'historique complet des modifications apportées à un projet de manière à pouvoir y revenir à tout moment.
- Collaboration : GitHub facilite la collaboration entre les membres d'une équipe de développement grâce à la possibilité de partager le code source et de travailler simultanément sur un même projet.
- Visibilité : GitHub permet de rendre un projet public, de sorte que d'autres développeurs peuvent le voir, le cloner et le modifier.
- Intégration : GitHub offre de nombreuses intégrations avec d'autres services, tels que les services de déploiement continu, les systèmes de suivi de bugs, les outils de gestion de projets, etc.
- Documentation : GitHub permet d'ajouter de la documentation à un projet pour faciliter la compréhension du code par les autres développeurs.
- Support communautaire : GitHub dispose d'une grande communauté de développeurs qui peuvent aider à résoudre des problèmes, proposer des améliorations ou fournir des conseils.
- Gratuité : GitHub offre des fonctionnalités de base gratuitement pour les projets open source, tandis que les fonctionnalités avancées sont payantes pour les projets privés.

En somme, GitHub est un outil incontournable pour tout développeur qui souhaite gérer efficacement le versioning de son code, collaborer avec d'autres développeurs et rendre son code plus visible et accessible à la communauté.

Dans ce contexte GitHub a permis un échange plus facile entre les parties prenantes de l'application et avoir des retours sur les avancées positives et négatives.

Voici le lien pour accéder au projet : [github.com/stephane-delire/SmartChrono-BC](https://github.com/stephane-delire/SmartChrono-BC)

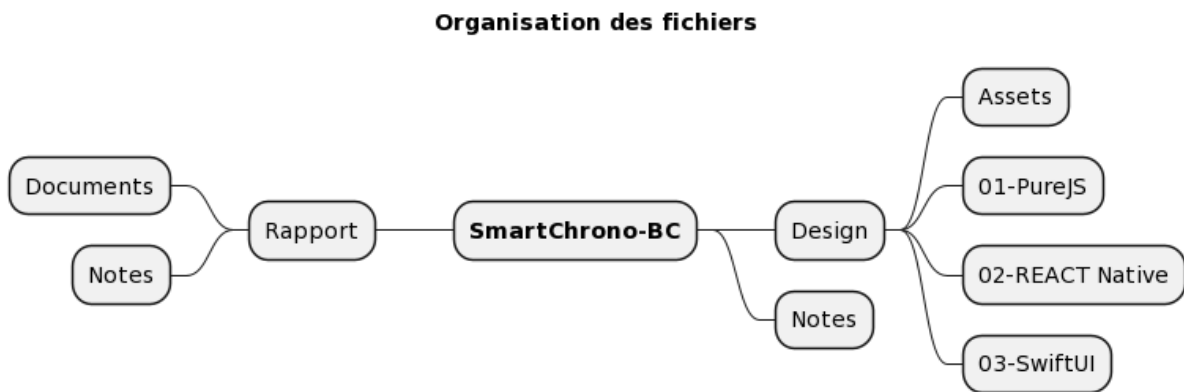
## Structure du projet

L'organisation des fichiers au sein du projet GitHub est clairement structurée. Le premier niveau de hiérarchie est constitué de la racine "SmartChrono - BC", qui contient trois dossiers distincts.

Le premier dossier, nommé "Design", comprend les trois versions de l'application qui ont été développées. La première version a été créée en Pure JS et était destinée à être compilée à l'aide d'Apache Cordova. La deuxième version a été développée en React Native et était destinée à être compilée avec React, mais elle pouvait également être testée en développement grâce à un compilateur directement installable sur smartphone, appelé "expo-go". Malheureusement, cette version a fini par ne plus pouvoir être compilée en raison de la complexité du programme. Enfin, la dernière version de l'application a été développée à l'aide de SwiftUI à partir d'un ordinateur Mac.

Au niveau supérieur de la racine, on trouve un dossier nommé "Notes" qui ne contient que des notes prises au fil du temps pour suivre l'avancement du projet.

Enfin, le dossier "Rapport" contient tous les fichiers nécessaires pour rédiger le travail.



En ce qui concerne le dossier "Documents", qui est dédié à la rédaction du rapport écrit, il contient tous les éléments visuels nécessaires, tels que des images et des diagrammes. Il est important de noter que le fichier Word principal a été créé à partir de plusieurs fichiers Word secondaires qui sont enregistrés dans le sous-dossier "Assets". Chaque sous-fichier est lié au fichier principal, ce qui facilite la rédaction et la mise en page.

## Langages « cross-platform »

Les langages cross-platform, également appelés langages de développement multiplateforme, sont des langages de programmation qui permettent de créer des applications qui peuvent être utilisées sur différentes plateformes et systèmes d'exploitation, tels qu'Android et iOS.

Voici quelques-uns des langages cross-platform les plus courants :

- **Flutter** : Flutter est un framework open-source développé par Google qui permet de créer des applications mobiles pour Android et iOS, ainsi que pour le web. Il utilise le langage de programmation Dart et offre une expérience de développement rapide et efficace.
- **React Native** : React Native est un framework open-source développé par Facebook qui permet de créer des applications mobiles pour Android et iOS en utilisant le langage de programmation JavaScript. Il utilise également la bibliothèque React pour la création d'interfaces utilisateur.
- **Xamarin** : Xamarin est une plateforme de développement multiplateforme appartenant à Microsoft qui permet de créer des applications pour Android, iOS et Windows en utilisant le langage de programmation C#. Il offre également des outils de développement avancés tels que des émulateurs et des outils de test.
- **Apache Cordova** : Apache Cordova est également un framework de développement d'applications multiplateformes open-source. Il est basé sur des technologies web telles que HTML, CSS et JavaScript, et permet de créer des applications mobiles pour plusieurs plateformes, notamment Android, iOS et Windows Phone.

Les avantages des langages cross-platform sont nombreux :

1. **Économie de temps et d'argent** : Avec les langages cross-platform, les développeurs peuvent créer des applications pour plusieurs plates-formes en utilisant un seul codebase, ce qui réduit considérablement le temps et les coûts de développement.
2. **Développement plus rapide** : Les outils de développement proposés par les langages cross-platform permettent une création rapide d'interfaces utilisateur et de fonctionnalités.
3. **Mise à jour plus facile** : Comme il n'y a qu'un seul codebase pour plusieurs plateformes, les mises à jour peuvent être effectuées simultanément pour toutes les plateformes, ce qui permet de gagner du temps et d'éviter les erreurs.
4. **Facilité de maintenance** : Avec une seule base de code, les mises à jour et les corrections de bugs peuvent être facilement appliquées à toutes les plateformes en même temps, ce qui simplifie la maintenance de l'application.
5. **Plus grande portée** : En développant une application cross-platform, on peut toucher un public plus large en la rendant disponible sur plusieurs plateformes.

Tout d'abord, j'ai opté pour le framework Cordova, car il permet de coder une application mobile comme une application web standard. J'ai développé une première version de l'application pour montrer à M. Burniaux l'avancement et pour vérifier si cela répondait à ses attentes. Cette première ébauche ne comportait qu'une interface utilisateur de base, sans aucune logique fonctionnelle, et pouvait être consultée depuis un navigateur web sur un ordinateur.

Développant sous Microsoft Windows j'ai pu remarquer que lorsque je souhaitais compiler le code grâce au framework Cordova, il me manquait un certains nombre de dépendance propre à Apple. Mr Burniaux me conseilla alors de me tourner vers un autre langage cross-platform, comme React Native qui a un module qui s'installe de base sur les appareils mobiles afin d'émuler un environnement stable.

Cela m'a amené à réécrire le code une deuxième fois, en vérifiant au préalable que je pouvais utiliser le module installé sur mon propre appareil mobile iOS pour émuler l'application. Au début, tout s'est bien déroulé : j'ai pu réécrire l'interface utilisateur grâce à React Native et commencer à implémenter un peu de logique fonctionnelle. Cependant, cela s'est compliqué lorsque cette logique est devenue plus complexe et que j'ai dû commencer à envoyer des requêtes HTTP vers le serveur Odoo. Malheureusement, le module n'était pas assez puissant pour exécuter mon code sur mon appareil, ce qui m'a ramené au point de départ.

De plus, si je souhaitais passer à une version native de l'application, c'est-à-dire une application compilée fonctionnant sans module complémentaire, React Native nécessitait les mêmes dépendances que Cordova pour Apple. Cela ne répondait finalement pas aux attentes requises.

### Langage Swift (Apple)

En définitive afin de mener ce projet à bien, j'ai décidé de changer mon environnement de travail complètement et de passer de Windows à Apple en travaillant sur un Mac. Ceci m'a permis d'être complètement en phase avec l'infrastructure Apple et de pouvoir avoir les outils de base proposés directement par Apple pour coder des applications iOS.

Il existe plusieurs langages de programmation pour iOS. Swift est un langage de programmation multiparadigme développé par Apple depuis 2014 pour le développement d'applications iOS, MacOS, WatchOS et TvOS. Il a été conçu pour être sûr, rapide et moderne, tout en étant facile à utiliser. Swift combine des éléments de langages de programmation impératifs, fonctionnels et orientés objet, et il est destiné à remplacer le langage Objective-C, utilisé historiquement pour le développement d'applications pour les plateformes Apple.

Swift est conçu pour être sûr en évitant les erreurs de programmation courantes telles que les pointeurs nuls ou les dépassements de tableau. Il offre également une syntaxe concise et expressive, ce qui le rend facile à apprendre et à utiliser. En outre, Swift est optimisé pour des performances élevées, ce qui signifie qu'il peut exécuter des tâches complexes rapidement, même sur des appareils avec des ressources limitées.

Swift est un langage de programmation fortement typé, ce qui signifie que chaque variable, constante et fonction a un type spécifique qui doit être déclaré explicitement. Cela permet à Swift de détecter les erreurs de type dès la compilation plutôt qu'à l'exécution, ce qui rend le code plus sûr et plus prévisible. En outre, la vérification de type aide les développeurs à comprendre comment utiliser correctement les différentes parties de leur code.

Swift offre également la possibilité de déclarer des types de données optionnels, qui peuvent avoir une valeur ou être nuls. Cela permet aux développeurs de gérer facilement les situations où une variable peut être absente ou avoir une valeur inconnue.

## Communications entre iOS et Odoo

Le principe est simple : l'application mobile doit être capable de communiquer via Internet en utilisant le protocole HTTPS pour envoyer des requêtes à un serveur Odoo qui répondra. M. Burniaux m'a mis à disposition un serveur de test Odoo pour que je puisse faire mes essais et coder ces requêtes.

Cependant, il était important que je comprenne comment le serveur Odoo communiquait avec les autres applications via son API. En consultant la documentation officielle d'Odoo, j'ai découvert qu'ils conseillaient l'utilisation du protocole XML-RPC.

J'ai essayé les quelques exemples présentés sur le site officiel d'Odoo en utilisant le langage Python, comme recommandé, mais j'ai constaté que ces exemples contenaient des erreurs.

En cherchant sur des sites spécialisés, j'ai finalement réussi à faire fonctionner ces appels en Python. Cependant, cela ne résolvait qu'une partie du problème, car j'utilisais du langage Python et non Swift, et la structure des données était au format XML, ce qui n'était pas idéal pour Swift.

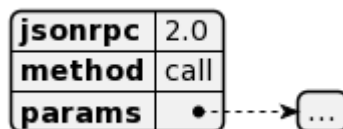
En dernier recours, j'ai examiné le code source d'Odoo (car c'est un logiciel open source) pour comprendre comment le serveur traitait les requêtes via son API, et j'ai découvert l'existence d'un protocole qui n'était expliqué nulle part dans la documentation : le JSON RPC.

Pour tester ce protocole, j'ai utilisé l'utilitaire en ligne Postman, qui permet d'envoyer des requêtes personnalisées et d'analyser les réponses.

En me basant sur le code source d'Odoo et sur la partie de la documentation qui était correcte, j'ai pu définir comment utiliser ce protocole.

La route pour y accéder est « /jsonrpc » juste après l'url de base d'un serveur Odoo et la méthode est en « post ». Les données à envoyer sont toujours sous format « .json » et se décrivent toujours comme ceci :

```
{"jsonrpc": "2.0", "method": "call", "params" : {...}}
```



Ceci constitue l'en-tête pour toute les requêtes vers le serveur Odoo. Les données situées sous la clé « params » sont les seules qui varient en fonction de la demande.

Par exemple si nous voulons récupérer les informations de base du serveur ce qui ne nécessite pas d'identification au préalable nous pouvons envoyer ceci :

```
{"jsonrpc": "2.0", "params": {"service": "common", "method": "version", "args": []}}
```





## L'identification d'un utilisateur

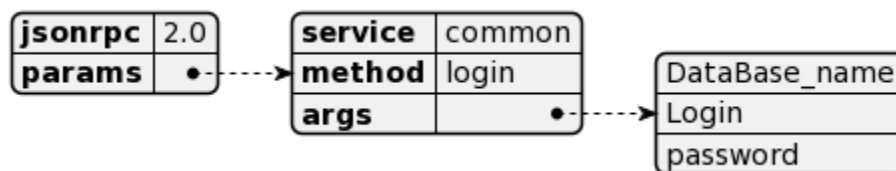
En général, un utilisateur dispose d'un nom d'utilisateur et d'un mot de passe pour se connecter à Odoo.

Cependant, nous verrons plus tard que les méthodes pour interagir avec Odoo ne demandent pas de nom d'utilisateur et de mot de passe, mais plutôt l'ID de l'utilisateur, qui est un nombre entier unique associé à chaque utilisateur.

La méthode de vérification consiste donc à obtenir cet ID à partir du serveur. Si le serveur ne renvoie pas l'ID, cela signifie qu'un des deux paramètres (nom d'utilisateur ou mot de passe) est incorrect lors de la tentative de connexion.

Les données à envoyer sont sous cette forme :

```
{"jsonrpc": "2.0", "params": {"service": "common", "method": "login", "args": ["DataBase_name", "Login", "password"]}}
```



Si les informations de noms de base de données, login d'utilisateur ainsi que de son mot de passe sont correctes le serveur renvoie à son tour une donnée sous format «.json » de ce type :

```
{"jsonrpc": "2.0", "id": null, "result": 0}
```

<b>jsonrpc</b>	2.0
<b>id</b>	null
<b>result</b>	0

On peut identifier dans les données reçues qu'il y a la présence de deux clés intéressantes : « id » dont la valeur est nulle, et la clé « result » dont la valeur vaut l'entier unique identifiant l'utilisateur dans le système d'Odoo.

Cet entier est important pour la suite car il faudra l'utiliser dans les requêtes plus avancées à la place du login qui sert d'identification. Ceci nous permet d'avoir une méthode d'identification simple mais sécurisée car utilisant le protocole « Https » et donc encrypté par SSL/TSL.

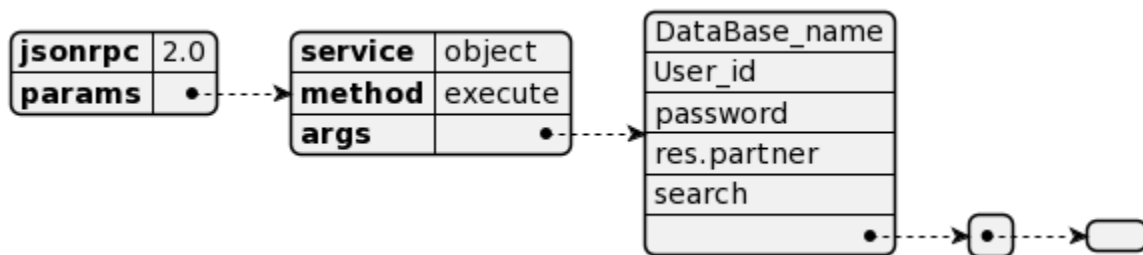
## Lecture d'une table de la base de données

Pour interagir avec la base de données d'Odoo il faut utiliser le service « object » et utiliser la méthode « execute ». Le code source montre que l'on peut également utiliser la méthode « execute\_kw ».

Attention qu'à partir de ce point, il ne faut donc plus utiliser la valeur de login pour l'utilisateur mais bien la valeur sous forme d'entier récupérée à l'étape précédente.

En terme d'exemple si je souhaite interroger la table « res.partner » qui est une table créée par défaut dans le système Odoo :

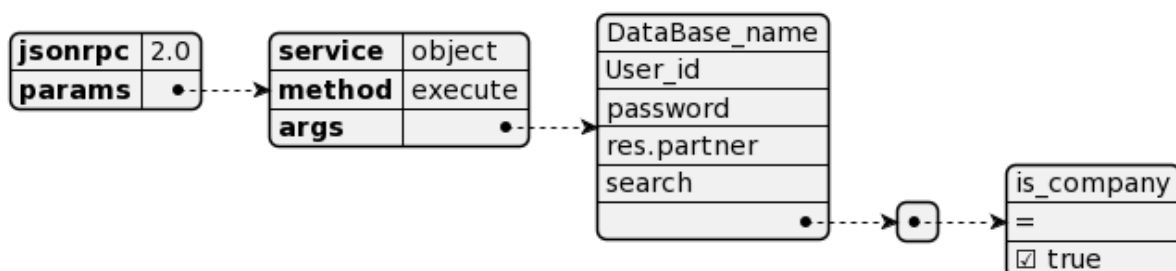
```
{ "jsonrpc": "2.0", "params": { "service": "object", "method": "execute", "args":
  ["DataBase_name", "User_id", "password", "res.partner", "search", [[]]] }
```



On remarque que sous la clé « args » se trouve un sous ensemble sous format liste d'arguments qui sert à définir l'action que l'on souhaite réaliser. L'avant dernier paramètre de la liste est dans cet exemple est la méthode « search » qui est utilisée pour afficher des enregistrements, elle est comparable à une requête SQL « SELECT ». A la suite de cette chaine de caractère « search » se trouve un nouveau sous ensemble formaté sous forme de liste. Ceci sert à paramétrer la méthode que l'on souhaite voir exécuter, dans le système Odoo cela s'appelle un « domain » et peut s'apparenter en requête SQL à la clause « WHERE ».

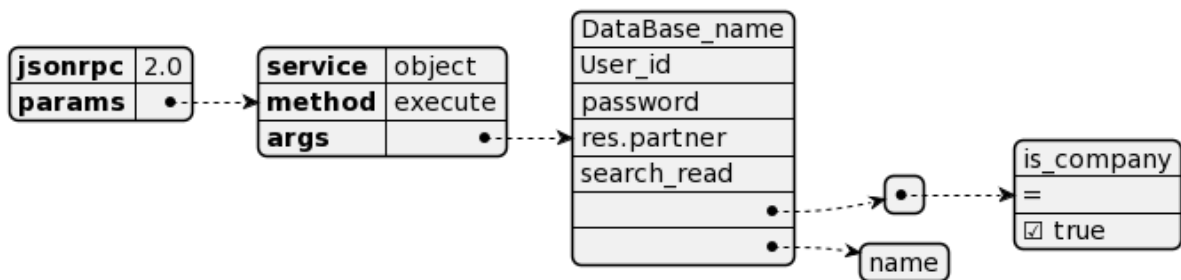
A titre d'exemple si je souhaite paramétrer ma requête et ainsi rajouter un domaine qui ne m'afficherait que les contacts qui sont une entreprise de mon carnet de contact :

```
{ "jsonrpc": "2.0", "params": { "service": "object", "method": "execute", "args":
  ["DataBase_name", "User_id", "password", "res.partner", "search", [{"is_company",
    "=", true}]] }
```



Ces derniers exemples me retournent la liste complète des colonnes présentes en base de données. L'équivalent en requête SQL serait « SELECT \* ». Ici je donne un dernier exemple incluant une requête vers le système Odoo en utilisant une méthode « search\_read » qui me permet de consulter les données et de les lire automatiquement, tout en filtrant si celles-ci sont d'une entreprise et en ne voulant lire que la colonne « name » de la table :

```
{"jsonrpc": "2.0", "params": {"service": "object", "method": "execute", "args":  
  ["DataBase_name", "User_id", "password", "res.partner", "search_read",  
    ["is_company", "=", true]], ["name"]}}
```



### Liste des méthodes possible pour le protocole JSON-RPC avec Odoo

Afin de mieux comprendre les échanges possibles entre un serveur Odoo et une application tierce (API), il est important de disposer d'une liste complète de ces méthodes. Cela permettra d'avoir une compréhension précise de l'architecture de l'application :

- **Search** : La méthode "Search" permet d'afficher une liste d'enregistrements et de les filtrer si nécessaire. Elle renvoie uniquement l'identifiant de ces enregistrements sous forme d'entier.
- **Search\_count** : La méthode "Search\_count" permet de renvoyer le nombre d'enregistrements présents dans une table spécifiée et de les filtrer si nécessaire (similaire à "Search").
- **Read** : La méthode "Read" permet de lire les données associées à un identifiant spécifié sous forme d'entier. Habituellement, cet identifiant est récupéré via la méthode "Search".
- **Fields\_get** : La méthode "Fields\_get" permet de récupérer uniquement les types de données qui définissent la ou les colonnes d'une table spécifiée. Des filtres peuvent être utilisés pour préciser quel type de colonnes doivent être incluses dans les informations renvoyées.
- **Search\_read** : La méthode "Search\_read" combine les méthodes "Search" et "Read" pour permettre une lecture directe des données en utilisant des filtres.
- **Create** : La méthode "Create" permet d'insérer de nouvelles données dans une table, similaire à l'opération "INSERT" en SQL. Si la requête est correcte, le serveur renvoie l'identifiant du nouvel enregistrement.
- **Write** : La méthode "Write" permet de mettre à jour des données déjà présentes dans une table, similaire à l'opération "UPDATE" en SQL.
- **Unlink** : La méthode "Unlink" permet de supprimer des données dans une table, similaire à l'opération "DELETE" en SQL.

## Réalisation du code informatique

### L'orienté objet

Un langage orienté objet est un type de langage de programmation qui se base sur le concept de l'« orientation objet ». Dans ce paradigme, les programmes sont structurés autour d'« objets », qui sont des entités regroupant à la fois des données et des méthodes pour manipuler ces données.

Les langages orientés objet permettent de modéliser des problèmes complexes en les divisant en objets individuels, représentant des entités du monde réel ou des concepts abstraits. Ces objets peuvent interagir entre eux en envoyant des messages, ce qui déclenche l'exécution de leurs méthodes spécifiques.

En utilisant le langage Swift d'Apple, les développeurs peuvent organiser leur code en classes et objets, encapsuler des données et des comportements dans des objets, hériter de classes existantes pour créer de nouvelles classes et utiliser des instances d'objets pour interagir avec le système et les autres objets. Swift offre donc un environnement solide pour la programmation orientée objet, tout en intégrant également d'autres paradigmes de programmation comme la programmation fonctionnelle.

### Réalisation des objets

Voici un aperçu visuel des objets qui ont été conçus pour l'application. Dans les prochains paragraphes, je détaillerai chacun d'entre eux.

C User
login : String password : String
init() loadUser() saveUser() reload()


C Settings
url : String db : String
init() loadSettings() saveSettings() reload()

C Project
projectData : [Int : String]
init() saveProjectData() flush() add(_key:Int, _value:String) reload()

C Task
taskData : [Int: [String: Int]]
init() saveTaskData() flush() add(_id: Int, _taskName: String, _projectId: Int) printData() reload()

C Record
filename : String
init() addRecord(date: String, id: String, duration: Int, project: Int, task: Int) count() saveData() flush()

### L'objet User

 User
login : String password : String
init() loadUser() saveUser() reload()

L'objet User a été créé dans le but de représenter les données de l'utilisateur de l'application. Il comprend trois informations : le login, le password et l'identifiant (non représenté dans l'image). Le login et le password sont des données demandées à l'utilisateur, sous forme de chaînes de caractères, et serviront à l'authentification de l'utilisateur sur le serveur Odoo. L'identifiant (id) est un entier (Integer) qui correspond à l'identifiant de l'utilisateur dans la base de données du serveur Odoo. Ce dernier sera renvoyé si les informations de login et de password sont correctes. Il est important de noter cet identifiant car il sera utilisé

ultérieurement pour interagir avec le serveur Odoo.


L'objet User comprend quatre fonctions. La fonction `init()` est automatiquement appelée lors de la création de l'objet et se contente d'appeler la fonction `loadUser()`.

La fonction `loadUser()` a pour rôle de charger un fichier appelé "user.json" à partir du dossier réservé par le système d'exploitation. Si ce fichier existe, la fonction tente de convertir ses données en un format de dictionnaire et les charge en mémoire pour les exploiter. Si le fichier n'existe pas ou si la fonction échoue lors du chargement des données, elle crée un nouveau fichier vide.

La fonction `saveUser()` a pour responsabilité de réaliser l'opération inverse de la fonction de chargement. Elle récupère les données présentes en mémoire concernant l'objet User, les convertit au format JSON, puis tente d'enregistrer ces données dans le fichier "user.json".

Enfin, la fonction `reload()` se charge de vider la mémoire en cours et tente de charger à nouveau le fichier "user.json".

### L'objet Settings

 Settings
url : String db : String
init() loadSettings() saveSettings() reload()


L'objet Settings a été créé dans le but de stocker les paramètres de l'utilisateur. Dans cette application, les paramètres ne se résument qu'à l'URL du serveur Odoo et au nom de la base de données. Ces données sont stockées sous forme de chaînes de caractères.

Parmi les fonctions de l'objet, on trouve `init()`, qui appelle la deuxième fonction `loadSettings()`. Celle-ci effectue une opération similaire à la fonction de chargement précédente, à ceci près que le fichier à charger est

"settings.json".

`saveSettings()` et `reload()` font la même chose également que dans le premier objet.

### L'objet Project

 Project
projectData : [Int : String]
init() saveProjectData() flush() add(_key:Int, _value:String) reload()


L'objet Project a été développé dans le but de représenter les projets stockés dans la base de données. Ces projets sont créés par un administrateur sur le logiciel Odoo et contiennent des tâches associées.

Cet objet représente uniquement le projet lui-même. Il est composé d'un identifiant (Integer) attribué automatiquement par le logiciel Odoo, ainsi que d'une chaîne de caractères représentant le nom du projet. Les projets sont ensuite stockés en mémoire sous forme d'un

dictionnaire (clé-valeur). Ce dictionnaire est stocké dans la variable "projectData".

Les fonctions init(), saveProjectData(), reload() et flush() sont identiques à celles des objets précédents. Cependant, la fonction add(key:Int, value:String) est plus complexe. Elle prend en paramètre l'identifiant de la variable d'entrée et le nom du projet afin d'ajouter correctement les nouvelles données au dictionnaire.

### L'objet Task

 Task
taskData : [int: [String:Int]]
init() saveTaskData() flush() add(_id:Int, _taskName:String, _projectId:Int) printData() reload()

L'objet Task a été développé pour représenter les tâches liées à des projets. Il est important de noter que ces tâches doivent contenir l'identifiant du projet auquel elles sont liées (jointure). Par conséquent, la structure du dictionnaire (clé-valeur) de la variable est légèrement différente par rapport à celle de l'objet Project. Dans ce cas, le dictionnaire est de la forme

suivante : [clé : [valeur : clé]], représenté dans l'objet en tant que "taskData".

La première clé représente l'identifiant unique de la tâche défini par le logiciel Odoo, qui est un entier (Integer). Sous cette clé, on trouve deux valeurs : le nom de la tâche, défini en tant que chaîne de caractères, et une autre clé définie également en tant qu'entier (Integer), représentant l'identifiant du projet auquel cette tâche est liée.

En termes de fonctions, l'objet Task présente les mêmes caractéristiques que les autres objets. La fonction add() demande en plus la variable représentant la deuxième clé liée au projet. Une fonction de test, printData(), est également présente. Cette fonction a été créée lors de la conception de l'application et sert à afficher dans la console de développement l'ensemble des données contenues dans la variable "taskData()".

### L'objet Record

 Record
filename : String
init() addRecord(date:String, id:String, duration:Int, project:Int, task:Int) count() saveData() flush()

Le dernier objet, Record, a été créé pour représenter les enregistrements de temps effectués par l'utilisateur via l'application. Une différence par rapport aux autres objets est que l'objet Record ne stocke pas en mémoire vive toutes les données des enregistrements. Il est plutôt appelé dynamiquement lorsque des données doivent être ajoutées ou supprimées.

L'objet Record ne contient qu'une variable, "filename", qui par défaut est définie comme "recordData.json" et représente le fichier contenant l'ensemble des données de temps.

Lors de l'exécution de l'application, lorsque des modifications doivent être apportées (ajout, suppression, etc.), l'objet charge dynamiquement le fichier à partir du répertoire, effectue les modifications nécessaires, puis sauvegarde le fichier.

En ce qui concerne les fonctions de cet objet, la plupart d'entre elles sont identiques à celles des autres objets. Il existe une fonction de test, "count()", qui permet d'afficher dans la console de développement le nombre d'enregistrements présents dans le fichier.

### Sauvegarde des données internes

Dans le cadre du développement de mon application iOS, un aspect primordial était la sauvegarde sécurisée des données conformément aux exigences spécifiées dans le cahier des charges. Pour répondre à cette exigence, j'ai utilisé l'objet FileManager de Swift, qui offre une solution fiable.

L'objet FileManager m'a permis de spécifier un "masque" sur les chemins d'accès aux fichiers. En utilisant ".userDomainMask" comme paramètre, j'ai pu garantir que les fichiers enregistrés par mon application seraient stockés dans un répertoire dédié, accessible uniquement par l'application elle-même.

Cette approche offre une protection supplémentaire en évitant d'enregistrer des données sensibles dans des emplacements accessibles par d'autres applications ou par l'utilisateur, tels que le dossier "Photos" ou autres.

De plus, il est important de noter qu'iOS ne dispose pas nativement d'un système de base de données intégré.

## Requêtes HTTPS

Le protocole HTTPS, qui signifie Hypertext Transfer Protocol Secure, est une version sécurisée du protocole HTTP utilisé pour le transfert de données sur Internet. La principale différence entre le HTTPS et le HTTP réside dans la sécurité des échanges de données.

Lorsqu'une connexion est établie via HTTPS, les données sont chiffrées à l'aide de protocoles de cryptage tels que SSL (Secure Sockets Layer) ou TLS (Transport Layer Security). Cela garantit que les informations échangées entre le navigateur ou une application de l'utilisateur et le serveur Web sont protégées et ne peuvent pas être lues ou modifiées par des tiers malveillants.

En plus du chiffrement, le HTTPS utilise également des certificats SSL/TLS qui permettent de vérifier l'identité du serveur auquel l'utilisateur se connecte. Ces certificats sont délivrés par des autorités de certification de confiance et assurent que l'utilisateur communique avec le véritable serveur et non avec un imposteur.

L'adoption du HTTPS est essentielle pour garantir la confidentialité des données sensibles telles que les informations de connexion et les données personnelles. Il offre une couche supplémentaire de sécurité et protège les utilisateurs contre les attaques telles que l'interception de données, le détournement de session et le vol d'informations.

Dans mon application, j'ai mis en place le protocole HTTPS pour assurer la sécurité des échanges de données entre l'application et le serveur. Cela garantit que toutes les interactions sont protégées et que les informations sensibles des utilisateurs sont sécurisées.

En l'absence d'informations détaillées dans la documentation officielle d'Apple concernant les requêtes HTTPS, j'ai recherché une solution alternative pour implémenter cette fonctionnalité dans mon application. J'ai découvert une bibliothèque open source appelée "Alamofire" qui répondait parfaitement à ces besoins.

Alamofire est une librairie largement utilisée et reconnue dans la communauté de développement iOS. Elle fournit une interface simple pour effectuer des requêtes réseau, y compris des requêtes HTTPS. En l'adoptant, j'ai pu bénéficier de fonctionnalités avancées telles que la gestion automatique des certificats SSL, la manipulation des en-têtes de requête et des réponses, ainsi que la gestion des erreurs.



## Code de l'interface utilisateur (SwiftUI)

En langage Swift, différents types de classes sont disponibles pour représenter des éléments à l'écran dans le développement de l'interface utilisateur. Par exemple, l'objet `TabView{}` permet de créer des boutons situés en bas de l'écran. Ces boutons peuvent être associés à des fonctions et des icônes, permettant ainsi de créer un menu de navigation (appelé Hub dans les chapitres précédents). Cependant, une vue ou un affichage en Swift n'est pas en soi un objet. Lorsque j'ai essayé de coder ces vues comme des objets, j'ai constaté que Swift ne proposait pas de mise à jour en temps réel. Les vues étaient calculées à l'avance et ne pouvaient pas être actualisées ultérieurement.

J'ai donc décidé de changer de paradigme et de passer à une approche plus procédurale. En utilisant des variables globales plutôt que des objets, et en adoptant un style de codage plus procédural, j'ai pu rendre mes vues plus dynamiques.

Par exemple voici un exemple de ce qu'on pourrait apparenter à un objet :

```
TabView{
    ViewChrono(record:record)
        .tabItem {
            Image(systemName:"timer")
            Text("Chrono")
        }
    ViewSync(record:record)
        .tabItem{
            Image(systemName:"icloud")
            Text("Sync")
        }
    ViewRecord(records:record)
        .tabItem{
            Image(systemName:"folder")
            Text("Record")
        }
    ViewHelp()
        .tabItem{
            Image(systemName:"questionmark.circle")
            Text("Help")
        }
}
```

Cet exemple illustre l'objet "`TabView{}`" qui m'a permis de réaliser le menu de navigation de l'application. On peut remarquer que chaque élément à l'intérieur correspond à un bouton que l'utilisateur peut interagir afin de naviguer entre les différentes vues. (Dans cet exemple respectivement les vues : Chrono, Synchronisation, Enregistrement et aide).

Afin d'illustrer un paradigme plus procédural voici un autre exemple qui concerne la vue d'authentification (première vue que l'utilisateur voit en lançant l'application). Il faut comprendre qu'il y a une variable globale nommée "loginState" définie en tant qu'entier (Integer) qui peut prendre plusieurs valeurs en fonctions des évènements :

```
@State var loginState: Int = 0
    // 0 : Login
    // 1 : Settings
    // 2 : Working Settings
    // 3 : Error saving settings
    // 4 : Settings recorded
    // 5 : Try To authenticate
    // 6 : Success
    // 7 : Error authentication
```

Chaque évènement, par exemple une clique sur un bouton, peut déclencher une modification de la vue. Dans ce cas, une fonction a été définie et appelée par cet évènement afin de modifier la vue :

```
//Switch view
func loginSwitchToMain() {
    loginState = 0
}
func loginSwitchToSettings() {
    loginState = 1
}
func loginSwitchToWorkingSettings() {
    loginState = 2
}
func loginSwitchToErrorSettings() {
    loginState = 3
}
func loginSwitchToSuccessSettings() {
    loginState = 4
}
func loginSwitchToTryAuthenticate() {
    user.login = userLogin
    user.saveUser()
    loginState = 5
}
func loginSwitchToSuccessAuthenticate() {
    loginState = 6
}
func loginSwitchToErrorAuthenticate() {
    loginState = 7
}
```

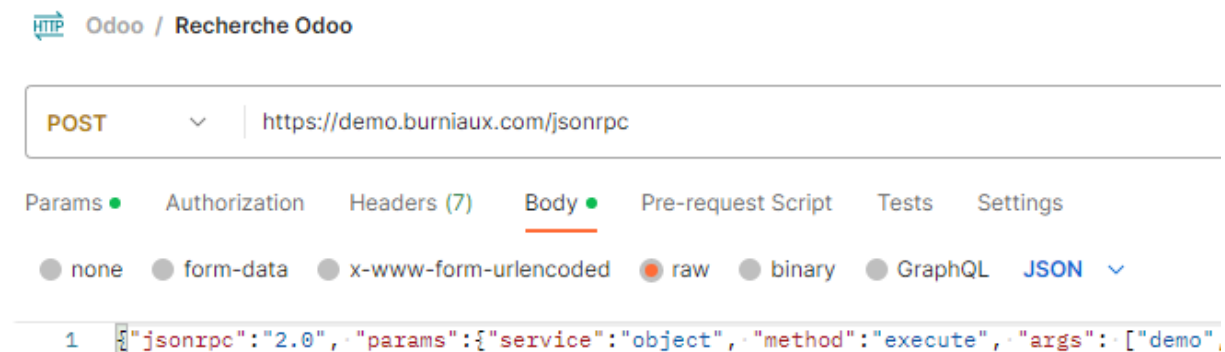
Voici un exemple de bouton, celui-ci permet de lancer la procédure d'identification :

```
Button(action: loginSwitchToTryAuthenticate) {
    Text("LOGIN")
        .fontWeight(.semibold)
        .foregroundColor(Color.white)
        .background(Color("Turquoise"))
        .frame(width: 315, height: 32)
        .overlay(
            RoundedRectangle(cornerRadius: 12)
                .stroke(Color.white, lineWidth: 0)
        )
}
```

## Test et validation

### Environnement de test Odoo et Postman

Comme mentionné dans un chapitre précédent, M. Burniaux a mis à disposition un serveur de test contenant une instance de l'ERP Odoo pour effectuer les tests nécessaires. Étant donné que la documentation officielle d'Odoo ne fournissait que peu d'informations sur l'utilisation d'Odoo en tant qu'API, j'ai donc consulté le code source de celui-ci. Afin de tester la route JSON-RPC, j'ai utilisé l'outil en ligne Postman ([web.postman.co](https://web.postman.co)) où j'ai pu examiner chaque méthode décrite dans un chapitre précédent en détail.



Postman est un utilitaire très populaire utilisé pour tester, développer et documenter des API. Il s'agit d'une application basée sur le cloud qui permet aux développeurs d'interagir avec des API en leur fournissant une interface conviviale et puissante.

Avec Postman, on peut envoyer des requêtes HTTP personnalisées à une API spécifique, que ce soit une requête GET, POST, PUT, DELETE, ou toute autre méthode prise en charge. On peut également spécifier des en-têtes personnalisés, des paramètres de requête, des données de formulaire ou du contenu JSON ou XML dans le corps de la requête.

## Environnement de test xCode

Xcode est un environnement de développement intégré (IDE) fourni par Apple pour les développeurs qui souhaitent créer des applications pour les appareils iOS, macOS, watchOS et tvOS. Il est spécifiquement conçu pour faciliter la création, le test et le déploiement d'applications sur les différentes plateformes d'Apple.

En ce qui concerne le développement d'applications iOS, Xcode est l'outil principal utilisé. Il offre un large éventail de fonctionnalités et d'outils pour faciliter le processus de développement. Voici quelques-unes des fonctionnalités :

- **Éditeur de code avancé** : Xcode propose un éditeur de code avec une coloration syntaxique, une complétion automatique, des suggestions de correction d'erreur, une navigation rapide dans le code, etc. Comme la plupart des IDE. Cependant il prend en charge les langages de programmation, Objective-C et Swift, utilisés dans le développement pour iOS.
- **Gestionnaire de projet** : Xcode facilite la gestion de projet en regroupant tous les fichiers source, les ressources et les dépendances dans une structure hiérarchique organisée.
- **Débogage et profilage** : Xcode propose des outils de débogage avancés tels que des points d'arrêt, des inspections de variables, des outils de suivi des appels de méthodes, etc. Il permet également de profiler les performances d'une application pour détecter les goulots d'étranglement et les problèmes de mémoire.

Concernant le simulateur d'iPhone intégré et de la capacité de compilation directe sur un téléphone Xcode comprend un simulateur d'iPhone intégré qui permet de tester une application sur différentes versions d'iPhone et d'iPad, directement depuis un Mac. On peut exécuter une application dans le simulateur pour évaluer son apparence, son comportement et sa compatibilité avec différentes tailles d'écran.

Concernant un téléphone réel, une fois que l'on a configuré un iPhone pour le développement, Xcode permet de compiler et d'exécuter directement une application sur un appareil. Cela permet de tester et de déboguer une application sur un matériel réel, ce qui peut révéler des problèmes spécifiques à l'appareil ou à l'environnement.

En ce qui concerne le déploiement d'une application iOS, il est important de noter que cela nécessite un compte de développeur professionnel chez Apple. Pour distribuer une application sur l'App Store, on doit s'inscrire au programme Apple Developer Program, qui est payant. Ce programme offre des avantages tels que l'accès aux dernières versions d'iOS, des ressources de support et la possibilité de soumettre des applications sur l'App Store.

Dans le cadre de ce travail de fin d'étude, je me suis arrêté à la compilation via Xcode sans aller jusqu'au déploiement sur l'App Store. Cependant, même sans déploiement sur l'App Store, on peut toujours tester l'application sur un appareil ou le partager avec d'autres personnes en utilisant des méthodes alternatives.

## Validation auprès de Mr Burniaux

Ayant pour professeur M. Burniaux, professeur lors du cours sur les logiciels ERP, et bénéficiant de cours dispensés le samedi, j'ai pu facilement prendre quelques minutes chaque semaine pour lui présenter mes progrès. Grâce à cette démarche, j'ai pu recevoir ses remarques, conseils et demandes, ce qui m'a permis d'avancer de manière significative. Sa disponibilité et son expertise m'ont aidé à avancer dans mon application.

## Futur de l'application

En ce qui concerne l'avenir de l'application, les plans de M. Burniaux doivent être pris en considération.

Il pourrait envisager de créer un compte développeur chez Apple afin de publier ou de vendre cette application, étant donné qu'elle est déjà fonctionnelle. De plus, il serait possible d'envisager l'ajout de fonctionnalités liées à Odoo.

À l'heure actuelle, l'application ne sert qu'à enregistrer le temps consacré à une tâche, mais Odoo dispose d'un vaste éventail de modules. Avec le code fonctionnel présenté ici, il serait relativement simple de transformer un smartphone en un outil avancé pour l'ERP Odoo. Par exemple, on pourrait imaginer une application capable de scanner des codes-barres pour vérifier les stocks ou de passer automatiquement des commandes auprès des fournisseurs.

En ce qui concerne le cadre actuel de l'application, il serait possible d'élargir ses fonctionnalités en permettant la création et l'édition de projets ainsi que la gestion des tâches associées, en plus de l'ajout de fonctionnalités de prise de notes.

## Conclusion

### Récapitulation des objectifs

Dans le cadre de mon travail de fin d'études, j'ai été confronté au défi de trouver un projet réel pour une entreprise. Grâce à la collaboration avec l'entreprise de M. Burniaux, j'ai pu identifier une opportunité de développement d'une application mobile. L'objectif principal de ce projet était de créer une application installable sur les smartphones, qui se connecte à l'ERP Odoo.

L'application mobile réalisée est un chronomètre de temps qui permet aux utilisateurs de comptabiliser leur temps sur les différentes tâches des projets présents dans le logiciel Odoo. Pour atteindre cet objectif, l'application doit établir une connexion avec l'ERP Odoo, vérifier les identifiants des utilisateurs pour garantir leur authentification, télécharger la liste des projets configurés par un responsable, et proposer une interface conviviale pour enregistrer le temps passé sur chaque tâche.

Une fois que les données de temps sont enregistrées dans l'application, celles-ci doivent être envoyées de manière sécurisée vers l'ERP Odoo pour une intégration fluide avec les fonctionnalités existantes. Ce processus de synchronisation permettra aux responsables de projet et aux autres utilisateurs de consulter les informations de suivi du temps.

## Résumé des résultats

Dans ma recherche de trouver une solution pour créer une application multiplateforme, j'ai découvert l'existence des langages de développement cross-platform, conçus pour s'exécuter sur différents environnements. J'ai alors décidé d'explorer React Native, un de ces langages, dans l'espoir de réaliser mon application mobile.

Cependant, j'ai rapidement rencontré des limites dans l'utilisation de React Native, notamment en termes de compilation du code. Je me suis rendu compte que pour compiler mon application, il était nécessaire d'utiliser un ordinateur Mac. Plutôt que de contourner cette contrainte en utilisant une solution alternative, j'ai pris la décision d'acquérir un Mac afin de pouvoir développer directement avec le langage approprié.

C'est ainsi que j'ai pu me familiariser avec le langage Swift et réaliser l'application de A à Z, en répondant aux attentes de M. Burniaux. Ce choix m'a permis d'exploiter pleinement les fonctionnalités et les performances de l'environnement iOS.

L'un des défis auxquels j'ai été confronté lors du développement de cette application était le manque de documentation officielle pour Odoo, l'ERP utilisé pour récupérer les données des projets. Cependant, grâce à la nature open-source du code source d'Odoo, j'ai pu analyser ce dernier et découvrir les mécanismes des API, ou les liaisons externes, permettant la communication avec l'ERP.

## Contribution à la connaissance

Ce projet m'a permis d'approfondir mes connaissances en développement d'applications mobiles et de mettre en pratique mes compétences en recherche et en analyse. En effet, lors du cursus scolaire à l'IETC-ps nous n'avons jamais vu de développement mobile.

Le manque de documentation m'a demandé de chercher des informations et des réponses là où la documentation traditionnelle faisait défaut. Cette expérience m'a permis d'améliorer mes compétences en recherche et en analyse de code source, ainsi que ma capacité à trouver des solutions créatives dans des situations où les ressources documentaires sont limitées.

Ce travail de fin d'études m'a offert l'opportunité d'explorer différentes technologies de développement, de passer du langage React Native à Swift pour réaliser l'application. Bien que le langage React Native ne m'a pas été nécessaire comparativement au langage Swift, celui-ci fait partie des nouveaux langages de programmation Web (principalement React) et est très utilisé à l'heure d'aujourd'hui, ce qui en fait un atout indéniable.

## Limitations et recommandations futures

Au cours de mon travail de fin d'études, j'ai rencontré certaines limitations qui ont influencé mes décisions et le développement de mon application mobile. La principale limitation était liée à l'environnement Apple, ce qui m'a conduit à l'achat d'un Mac pour pouvoir développer avec le langage Swift. Cette décision a été motivée par le fait que j'ai lu à plusieurs reprises lors de mes recherches que cela était nécessaire, mais en rétrospective, j'aurais peut-être dû explorer davantage les autres options, telles que la création d'une machine virtuelle avec un système d'exploitation Apple pour la compilation.

Une autre limitation majeure a été le compte développeur payant d'Apple. Pour pouvoir installer l'application sur un appareil Apple, il est nécessaire de connecter physiquement un iPhone ou tout autre appareil à mon Mac. Bien que cela ait été suffisant pour les besoins de mon travail de fin d'études, cela aurait été une limitation importante si j'avais développé une application destinée à une utilisation réelle. L'obligation d'un compte développeur payant peut restreindre la distribution de l'application.

Ces limitations soulignent l'importance de bien évaluer les contraintes techniques et les coûts associés lors du choix de la plateforme de développement et des langages. Dans le cadre de ce projet, j'ai rapidement adopté Swift sans explorer suffisamment les alternatives cross-platform. Une approche plus approfondie de ces solutions aurait pu offrir des alternatives plus flexibles et économiques.

En conclusion, bien que j'aie réussi à développer une application mobile fonctionnelle répondant aux attentes de mon travail de fin d'études en utilisant Swift, j'ai rencontré des limitations significatives liées à l'environnement Apple et au compte développeur payant. À l'avenir, il serait important d'élargir mes connaissances sur les options cross-platform et d'explorer plus en profondeur les différentes possibilités techniques afin de prendre des décisions plus éclairées lors du choix des langages et des plateformes de développement. Cela permettrait de maximiser l'efficacité et la flexibilité du développement tout en minimisant les contraintes et les coûts associés.

## Conclusion générale

Je suis satisfait d'avoir pu répondre aux attentes de M. Burniaux en réalisant une application fonctionnelle et en respectant les exigences du projet. Ce travail a été une expérience enrichissante qui a renforcé ma confiance en tant que développeur et m'a préparé pour de futurs défis dans le domaine du développement d'applications.

## Remerciements

La réalisation de ce travail de fin d'étude a été possible grâce au concours de plusieurs à qui je voudrais témoigner toute ma gratitude.

Je voudrais dans un premier temps remercier, mon directeur de suivi de ce travail Mr François Burniaux, professeur de progiciel de gestion intégré, pour sa patience, sa disponibilité, sa bonne humeur sans fin et surtout ses judicieux conseils. Sans lui, je me retrouvais sans sujet pour cette fin d'année, il a su être présent pour les élèves qui comme moi, s'était retrouvé sans sujet.

Ensuite, j'aimerais remercier Mr Mbayo Alain que j'ai eu comme professeur sur les bases de données avancées et en projet d'intégration et de développement et qui a tenu des sessions le samedi après-midi afin de nous aider à réaliser ce travail de fin d'étude.

J'aimerais également remercier Mr Druine Rudy et Mr El Jasouli Yasser, dans un premier temps pour m'avoir transmis leurs connaissances mais surtout d'avoir été présent pour proposer des sujets de travail pour cette fin d'année scolaire.

Je remercie également toute l'équipe pédagogique de l'IETC-ps de Charleroi et les intervenants professionnels responsable de ma formation, pour avoir assuré la partie théorique de celle-ci.



## Webographie et documentations

Concernant l'entreprise Burniaux Consulting de Mr Burniaux :

<https://www.burniaux.com/>

Concernant Odoo, qui est l'ERP auquel l'application se connecte :

[https://www.odoo.com/fr\\_FR](https://www.odoo.com/fr_FR)

Et sa documentation, même si incomplète :

<https://www.odoo.com/documentation/15.0/fr/>

Lien vers un site de contenu sur Odoo (blog, tutoriel, aide, ...) :

<https://www.cybrosys.com/>

Le lien Git-hub vers le code source d'Odoo :

<https://github.com/odoo/odoo>

Concernant le langage Swift, qui est le langage de programmation utilisé pour réaliser l'application :

<https://developer.apple.com/develop/>

Tutorial sur SwiftUI :

<https://www.hackingwithswift.com/quick-start/swiftui>

Concernant le langage React-native, qui est l'un des langages utilisés pour les premières versions de l'application :

<https://reactnative.dev/>

Tutorial sur React-native :

[https://www.tutorialspoint.com/react\\_native/react\\_native\\_overview.html](https://www.tutorialspoint.com/react_native/react_native_overview.html)

[https://www.youtube.com/watch?v=0-S5a0eXPoc&ab\\_channel=ProgrammingwithMosh](https://www.youtube.com/watch?v=0-S5a0eXPoc&ab_channel=ProgrammingwithMosh)

Ensuite Cordova :

<https://cordova.apache.org/>

Stackoverflow, qui est un site web qui rassemble une grande communauté de développeur proposant une plateforme d'échange sous forme de questions/réponses :

<https://stackoverflow.com/>

Lien vers le repository Git-hub où est hébergé l'application :

<https://github.com/stephane-delire/SmartChrono-BC>

Les illustrations ont été réalisées grâce au logiciel Adobe Photoshop, et les schémas et diagrammes ont été réalisés grâce à l'application en ligne PlantUML :

[https://www.adobe.com/be\\_fr/products/photoshop](https://www.adobe.com/be_fr/products/photoshop)

<https://plantuml.com/fr/>

## Annexes

{A faire...}