CoronaAlert

Vision

Version 1.0



Historique des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Auteur** |
| <17/05/2020> | 1.0 | Création du document | Taha Lekhlaikh |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table des matières

1. Introduction 4

2. Objectif du logiciel 4

2.1 Contexte 4

2.2 Besoins fonctionnels 4

2.3 Besoins non fonctionnels 4

3. Structure 4

3.1 Vue des couches 4

3.2 Sous-systèmes et paquetages 4

3.3 Interfaces 5

4. Comportement 5

4.1 Réalisation des cas d’utilisation 5

4.2 Mécanismes 5

5. Autres vues 5

5.1 Vue processus (optionnel) 5

5.2 Vue implémentation (optionnel) 5

5.3 Vue déploiement (optionnel) 6

5.4 Vue données (optionnel) 6

6. Concepts du domaine 6

7. Qualités de l’architecture 6

8. Points ouverts 6

Document d'Architecture du Logiciel

# Introduction

Le but du document est de décrire l’architecture du logiciel de notre projet CoronaAlerte concernant les différents modules qu’englobe ce dernier. La structure suivante permet de garantir que la conception réalisé répond aux besoins du système et que l’architecture en conséquence est maintenable et extensible.

Ce document met en collaboration tous les membres de l’équipe : architectes, concepteurs…

# Objectif du logiciel

Afin de contribuer à l’éradication de l’épidémie Covid’19, l’objectif est de réaliser une application de tracking et d’accompagnement des citoyens.

Le projet a été lancée par l’Etat marocain début Avril 2020. Une telle application a déjà été développée par plusieurs pays notamment la Corée du sud, la Chine, le Ghana. Des ressources ont été partagées en open source et peuvent être utilisées pour optimiser le projet. Le respect des données personnelles (Loi n° 09-08 relative à la protection des personnes physiques à l’égard du traitement des données à caractère personnel) doit être pris en compte.

## Contexte

## La Covid-19 est une maladie infectieuse qui a touché plus de 4 millions personnes, causant la mort de plus 270 millions et ne cesse d’augmenter dans certaines pays malgré l’enregistrement d’un nombre record de guérisons

## Besoins fonctionnels

## prévenir immédiatement si vous avez été en contact rapproché avec une personne que vous ne connaissez pas et qui vient d’être testée positive au COVID-19

## Si une personne est testé positivement , alerte l’ensemble des personnes utilisatrices de l’application avec qui a été en contact rapproché dans les derniers jours

## Besoins non fonctionnels

## L’application doit rappeler les consignes à respecter et les méthodes préconisées (lavage des mains, porte de masque, …)

## L’application doit permettre de connaitre l’état de l’épidémie dans le pays

# Structure

[Cette section décrit la structure statique de la vue logique de l’architecture, en montrant les composants, leurs interconnexions et les interfaces offertes par ces composants.

Présenter le style d’architecture utilisé : frameworks, patterns d’architecture...

Montrer éventuellement les alternatives envisagées pour l’architecture et dire pourquoi elles n’ont pas été retenues.]

## Vue des couches

[diagramme qui présente les couches du logiciel. Le rôle de chaque couche est explicité et la présence de chaque couche est justifiée.]

## Sous-systèmes et paquetages

[si le logiciel est complexe, faire un diagramme pour chaque couche qui montre les sous-systèmes ou paquetages qu’elle contient. Sinon faire un seul diagramme avec tous les sous-systèmes ou paquetages et leur dépendances.

Présenter le rôle de chaque sous-système ou paquetage, avec une courte description.

Indiquer clairement si le composant est réutilisé tel quel, adapté ou développé.

Présenter les classes importantes du paquetage.]

## Interfaces

[optionnel. Description de chaque interface avec ses opérations.]

# Comportement

[Cette section montre comment la collaboration des composants répond aux besoins fonctionnels et non fonctionnels.]

## Réalisation des cas d’utilisation

[pour chaque cas d'utilisation identifié comme important pour l’architecture, présenter sa réalisation. On prendra un ou plusieurs scénarios du cas d’utilisation et on le décrira par un diagramme d’interaction. Les objets apparaissant dans les diagrammes doivent être clairement identifiés dans la partie structure. Ajouter si nécessaire des explications pour qu’on comprenne bien comment les éléments structurels contribuent au scénario.]

## Mécanismes

[Présenter les mécanismes et patterns du logiciel.

Pour chacun décrire son mode d’emploi pour qu’un concepteur puisse l’utiliser facilement. Ajouter éventuellement des diagrammes d’interaction. Tracer avec les besoins non fonctionnels.]

# Autres vues

## Vue processus (optionnel)

[Cette section présente les classes actives (processus et “threads”) du logiciel, et montre le “mapping” avec les composants de la vue logique. Décrire les principaux mécanismes de communication : envoi de messages, interruptions, signaux, files d’attente …]

## Vue implémentation (optionnel)

[Cette section décrit l’organisation des composants physiques (code) et le mapping avec les composants logiques.]

## Vue déploiement (optionnel)

[Cette section décrit la partie matérielle de l’infrastructure et montre le mapping des composants physiques ou logiques ou des processus sur les nœuds. Chaque nœud et les interconnexions entre eux sont décrits.]

## Vue données (optionnel)

[Cette section présente les données persistantes.]

# Concepts du domaine

[Une description des concepts spécifiques du domaine et de leurs relations. Préciser comment les choix d’architecture préservent l’indépendance de ces concepts vis à vis de la technologie.]

# Qualités de l’architecture

[Avantages et inconvénients de l’architecture. Limitations en vue d’une extension.]

# Points ouverts