Project planning

MaRoWo Case 2

***Gemaakt door:***

*Max Mommersteeg*

*Rob Gerritsen*

*Wouter Fennis*

***Datum:*** *01-12-2016*

Inhoud

[Scope 3](#_Toc468365105)

[Sprintduur 3](#_Toc468365106)

[Teamafspraken 3](#_Toc468365107)

[(Generic) Definition of Done 3](#_Toc468365108)

[Vergaderingen 3](#_Toc468365109)

[Scrum Poker 3](#_Toc468365110)

[Scrum Board 4](#_Toc468365111)

[Overige afspraken 4](#_Toc468365112)

[Teamleden 4](#_Toc468365113)

[Risico’s 5](#_Toc468365114)

[RDW APK Server 5](#_Toc468365115)

[.NET Core 5](#_Toc468365116)

[Docker 5](#_Toc468365117)

[Auditlog 5](#_Toc468365118)

[Tools & Frameworks 6](#_Toc468365119)

[Doen we TDD? 6](#_Toc468365120)

[Gebruiken we MoQ? 6](#_Toc468365121)

[Gebruiken we Entity Framework Core? 6](#_Toc468365122)

[Gebruiken we EasyNetQ? / Onze eigen bus-implementatie? 6](#_Toc468365123)

[Wat moeten we nog meer afstemmen over tools & frameworks? 6](#_Toc468365124)

# Scope

Het te bouwen systeem moet kunnen communiceren met de RDW APK Server voor het indienen van APK aanvragen. Het daadwerkelijk indienen van deze aanvragen gaat de RDW APK Server doen.

Via het systeem moeten onderhoudsopdrachten kunnen worden toegevoegd. Daarnaast moet het mogelijk zijn om klanten aan deze onderhoudsopdrachten te kunnen koppelen.

# Sprintduur

Sprints duren gemiddeld genomen 2 dagen. Met 2 dagen tussen iedere oplevering, verwachten wij voldoende demo’s te kunnen geven. Daarnaast spreken we de product owner frequent genoeg om genoeg feedback te ontvangen. Wel moet er rekening worden gehouden met de beschikbaarheid van de product owner. Aangezien het project 6 dagen duurt zijn dat dus 3 sprints totaal.

# Teamafspraken

## (Generic) Definition of Done

* Altijd comment block toevoegen bij publieke en ingewikkelde (te ingewikkeld blijkt uit code review) methoden.
* Code reviewing mag je niet zelf doen
* Testen moeten slagen voor pull-request
* Ingewikkelde / foutgevoelige code wordt getest (blijkt uit code review)
* Pull-request gebruiken. Niet direct pushen naar master
* Side-effects beperken
* Onvermijdelijke Side-effects documenteren met commentaar

## Vergaderingen

* Daily scrum om 09:00 iedere dag.
* 2 uur vóór oplevering Backlog refinement
* Sprint Review op 24-11-2016, 29-11-2016, 01-12-2016 op het eind van de middag
* Sprint Planning direct na Sprint Review
* Retrospective direct na Sprint Planning

## Scrum Poker

We doen een soort van planning poker. Echter gebruiken we hierbij geen officiële poker kaarten.   
Fibonacci: 1, 2, 3, 5, 8, 13 als keuzes. (Daarboven is te groot. Verklein User Story)

## Scrum Board

We gebruiken naast een fysiek board in het lokaal ook een digitaal board (Jira).  
Het digitale board wordt vooral gebruikt voor backlog beheer en het fysieke board voor de actuele sprint.

## Overige afspraken

* Bij ziekte of vertraging, melden via Telegram
* Bij terugkoppelen van Pull-request melding geven over problemen via Github
* Max merged code naar Master
* Altijd pushen voor je naar huis gaat. Ook bij Work In Progress
* Branches noemen naar: us-mar-[userstorynr]

## Teamleden

|  |  |
| --- | --- |
| Teamlid | Intresse |
| WOUTER FENNIS | * Front-End (JavaScript, HTML, CSS) * Code reviewing * Eventing * TDD |
| Rob Gerritsen | * API * Back-End * Database * Docker * Bug Fixing |
| Max Mommersteeg | * Back-End * Front-End * Reviewing * Bug Fixing |

# Risico’s

## RDW APK Server

Om APK aanvragen in te kunnen dienen bij de RDW is een koppeling met de RDW APK server vereist. De product owner heeft ons voorzien van documentatie. Deze documentatie beschrijft hoe van de server gebruik kan worden gemaakt. Daarnaast worden er voorbeelden van output getoond. Echter worden hiermee niet alle mogelijke scenario’s afgedekt.

Na het bekijken van de documentatie zijn er een vragen naar boven gekomen:

* Welke reactie kan worden verwacht bij een foutief keuringsverzoek?
* Hoe ziet een keuringsverzoek antwoord eruit, wanneer er geen steekproef is?

## .NET Core

Het gebruik van .NET Core is een vereiste voor dit project. Echter is bij het team bekend dat .NET Core problemen kan geven. Toch denken wij als team de gewenste functionaliteit te kunnen realiseren. Deze conclusie hebben wij kunnen trekken aan de hand van het eerste gesprek met de product owner en documentatie op Google Drive, beschikbaar gesteld door de product owner.

Onderstaande punten kunnen zorgen voor vertraging:

* Problemen met het ophalen van referenties. Project.json corrupt
* Documentatie niet volledig

## Docker

De systeembeheerders van Marowo wensen de opgeleverde software eenvoudig in productie te kunnen zetten door middel van Docker. Het team beschikt over de nodige kennis met betrekking tot Docker. Echter weet het team nog niet hoe meerdere applicaties aan elkaar gelinkt kunnen worden met behulp van Docker. Hoe dit werkt zal moeten worden uitgezocht.

## Auditlog

Wanneer een van de servers offline gaat, is het van belang dat deze up-to-date wordt gebracht wanneer deze weer online komt. Alle berichten moeten worden gepersisteerd. Deze berichten moeten later bij online komen van de server kunnen worden uitgelezen.

Voor dit project gaan we gebruikmaken van een eigen Eventbus implementatie. Alle Auditlog berichten worden gepersisteerd, echter is er geen functionaliteit voor het uitlezen van deze AuditLog. Wanneer de product owner dit eist of wenst, moet deze functionaliteit tijdens de loop van het project worden geschreven. Dit resulteert in meer werk.

# Tools & Frameworks

**Verplicht: RabbitMQ, Docker, ASP.NET Core**

## Doen we TDD?

TDD werken is geen must. Zoals eerder beschreven in de Generic Definition of Done.

## Gebruiken we MoQ?

Gebruik van zowel handgeschreven mocks als mocks geschreven met behulp van MoQ. Programmeur maakt keuze op basis van test scenario.

Gebruiken we Entity Framework Core?

Ja, we maken gebruik van SQL Server, we communiceren hiermee met behulp van Entity Framework Core.

## Gebruiken we EasyNetQ? / Onze eigen bus-implementatie?

We gaan gebruik maken van de bus-implementatie van Rob. Deze is nog niet volledig functioneel en zal tijdens de case nog wat werk vereisen.

## Wat moeten we nog meer afstemmen over tools & frameworks?

Voor het bijhouden van de Backlog maken we gebruik van Jira. De sprint backlog staat wel op het fysieke scrumboard.

We gaan gebruik maken van Github en dus niet van TFS. Laatstgenoemde gebruiken we enkel om de code aan het eind van de case te persisteren op de TFS server.

Voordeel is dat Github altijd te benaderen is en de TFS server enkel op locatie in Utrecht. Daarnaast ook mogelijkheid tot integratie met Telegram.

Programmeur kiest een Git client naar keuze. We maken gebruik van de Pull-Request functionaliteit van Github. Daarnaast kan de review functionaliteit worden benut voor het geven van terugkoppeling op Pull-Requests. Tot slot maken we gebruik van ‘Github Issues’.