# Ontwikkeling Smartphone/tablet applicatie

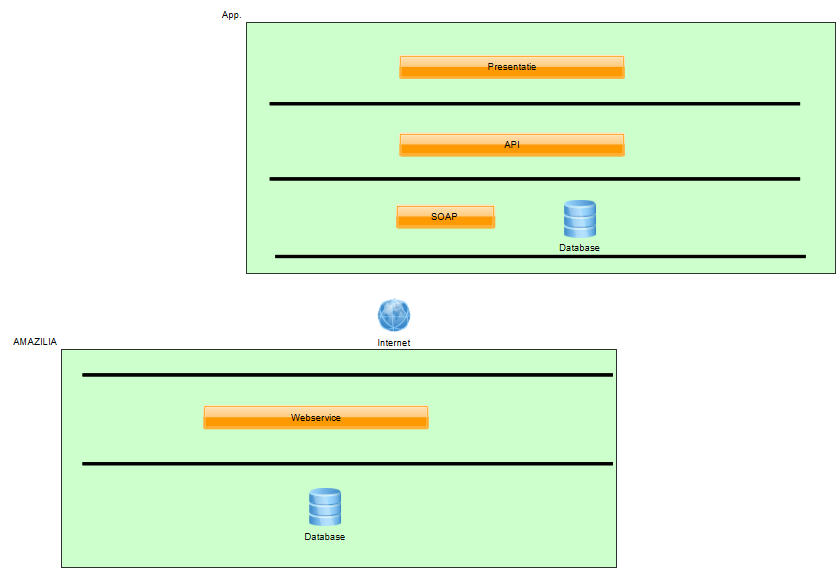
Dit document omschrijft in het kort de technische werking van een te ontwikkelen app. Doel van dit document is om een goede tijdsinschatting te maken.

Technisch beschreven:

* We maken gebruik van de phonegap/telerik mogelijkheden. Phonegap is platformafhankelijk, Dit houd in dat een gemaakte app bruikbaar is voor bijvoorbeeld de ipone en android.
* Het skelet wordt opgebouwd doormiddel van statische html.
* Onder het skelet verstaan we:
  + Tabbladen
  + De opbouw van de blokken
  + De zwart omrande teksten/afbeeldingen in onderstaande screenshot
  + Menu
* Er wordt gebruik gemaakt van jquery mobile, dus templates kunnen worden in gezet: <https://www.template.net/design-templates/web-elements/40-best-jquery-mobile-templates/>
* De blokken worden gevoed vanuit het CMS, hiervoor worden catalogus modules ingezet. De in te zetten modules worden later beschreven.
* De App. Zal beschikken over een eigen database, waar de berichten en instellingen in opgeslagen worden.
* De App kan meertalig worden gemaakt, de taal wordt initieel bepaald aan de hand van de systeeminstellingen. De taal is later nog door de gebruiker in te stellen.

# 1.0 Api

Een api wordt ontwikkeld die de gegevens uit het CMS haalt en in de cache database plaatst. Doormiddel van de Api kunnen de berichten makkelijk opgehaald worden.



Korte beschrijving van de api:

CMS.catalogInit( Catalogs, fnComplete)

Catalogs: Een array van catalogus ID’s die opgehaald wordt.

fnComplete: Een callback functie die aangeroepen wordt indien voltooid.

De objecten worden opgehaald van de server en in de cache DB opgeslagen. Alleen de nieuw toegevoegde objecten worden opgehaald. Het ophalen van nieuwe objecten gebeurt tijdens opstarten, en na een bepaalde expiratie tijd b.v. 3 uur.  
Deze functie wordt aangeroepen bij het laden van een tabblad.

CMS.getObjects(CatalogID, CategoryID, [ObjectID], fnComplete)

CatalogID: De catalogus van toepassing

CategoryID: De categorie van toepassing

ObjectID : optioneel, het objectID van toepassing.

fnComplete: Een callback functie die aangeroepen wordt indien voltooid.

De objecten worden uit de lokale cache DB gehaald, database acties zijn altijd asynchroon, dus een callback functie is noodzakelijk. fnComplete wordt uitgevoerd, met als parameters de opgehaalde objecten.

De objecten kunnen als in onderstaand voorbeeld benaderd worden, en zo de sliders/html vullen.

fnComplete( e){  
 e.Objects[0].spec[‘img’];  
}

CMS.getDivContent(PageID, Code, fnComplete)

PageID: Pagina van toepassing

Code : Code van kader

fnComplete: Een callback functie die aangeroepen wordt indien voltooid.

Met deze functie wordt pagina inhoud opgehaald.

# 2.0 Module structuur

De content word deels gehaald uit bestaande modules, voor de overige App. specifieke onderdelen wordt er een App. catalogus aangemaakt met meerdere divisie specifieke categorieën.

**Corporate**:

* Events komt uit de al bestaande events Agenda
* Nieuws komt uit de bestaande nieuwsmodule, categorie: alle nieuws.
* Slider is een categorie uit de App. catalogus.
* Historie is een product uit de statische content categorie van Corporate.
* Rondleidingen is een product uit de statische content categorie van Corporate.

**Frontline Systems**

* Nieuws komt uit de bestaande nieuwsmodule, categorie Frontline Systems
* Events komt uit de al bestaande events Agenda, categorie Frontline systems
* Het bovenste blok is een product uit de statische content categorie van Frontline Systems uit de App. Catalogus

Voor de overige tabbladen geld hetzelfde als voor Frontline Systems

# 3.0 Urenoverzicht

|  |  |
| --- | --- |
| Omschrijving | Uren |
| **Back-end development** |  |
|  |  |
| Ontwikkeling Api | 16 |
| Ontwikkeling soap service | 6 |
| Database cache techniek | 8 |
| Inloggen | 8 |
| Meertalig | 4 |
|  |  |
|  |  |
| **Front-end development** |  |
|  |  |
| Implementatie template/design | 24 (afhankelijk van design) |
|  |  |
| **Design** |  |
|  |  |