<Autoverhuur-Systeem>

Versie 2.0

Documenthistorie

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Versie** | **Beschrijving** | **Auteur** |
|  | 0.1 | Initiële versie |  |
| 20-2-2013 | 1.0 | De eerste versie die ingevuld is. | Thom Trignol |
| 03-7-2013 | 1.1 | Reconstructie en aanvulling origineel | Julian West |
| 04-7-2013 | 2.0 | Verificatie en spellingscontrole | Thom Trignol |

Distributie

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Naam** | **0.1** | **1.0** | **1.1** | **2.0** |  |  |  |  |  |  |  |  |  |  |
| Thom Trignol |  | x | x | x |  |  |  |  |  |  |  |  |  |  |
| Julian West |  | x | x | x |  |  |  |  |  |  |  |  |  |  |
| Paul Lidnelauf |  | x |  | x |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Accordering document

|  |
| --- |
| Namens  Ondergetekende is nog niet gespecificeerd.  ………………… |

|  |
| --- |
| Namens <Julian&Thom Cooperation>  Julian West & Thom Trignol |

Inhoudsopgave

[1. Inleiding 4](#_Toc360713091)

[1.1 Doel van het system 4](#_Toc360713092)

[1.2 Design goals 4](#_Toc360713093)

[1.3 Definities, acroniemen en afkortingen 4](#_Toc360713094)

[1.4 Referenties 4](#_Toc360713095)

[1.5 Overzicht 4](#_Toc360713096)

[2. Current Software architecture 5](#_Toc360713097)

[3. Voorgestelde software architecture 6](#_Toc360713098)

[3.1 Overzicht 6](#_Toc360713099)

[3.2 Subsystem decomposition (UML class diagrams) 6](#_Toc360713100)

[3.2.1 Architectuurstijl 7](#_Toc360713101)

[3.2.2 Concurrency 7](#_Toc360713102)

[3.3 Hardware/software mapping (UML deployment diagrams) 7](#_Toc360713103)

[3.4 Persistent data management 7](#_Toc360713104)

[3.5 Access control en security 7](#_Toc360713105)

[3.6 Global software control 8](#_Toc360713106)

[3.7 Boundary conditions 8](#_Toc360713107)

[3.7.1 Start up 8](#_Toc360713108)

[3.7.2 shutdown 8](#_Toc360713109)

[3.7.3 Foutafhandeling 8](#_Toc360713110)

[4. Subsysteem services 9](#_Toc360713111)

# Inleiding

## Doel van het system

Het systeem moet uit opdracht van 'On the road again with us' de administratie van auto's op filialen van hun bedrijf deels automatiseren. Hieronder valt de verhuur, verkoop en intern transport van auto's. Ook zouden vaste en niet-vaste klanten moeten worden gemanaged.

## Design goals

* Beveiliging - Het systeem moet rechten verlenen alleen aan bevoegde mensen. Dit zou preventief moeten werken tegen hackers en zorgen dat een andere afdeling niet per ongeluk aanpassingen maakt in het onderdeel van een andere afdeling. Zie SDD par. 3.5
* Specificatie - Het systeem zal gespecificeerd worden in UML, voor duidelijkheid met de werkgever en programmeurs. De werkgever kan meer invloed hebben op het eindproduct en de programmeurs zullen beter kunnen samenwerken. Zie RAD, ODD en dit SDD.
* Gebruiksvriendelijkheid - Het systeem moet door medewerkers intuïtief gebruikt kunnen worden. Dit is voor het design belangrijk. Zie RAD hoofdstuk 4.
* Uitbreidbaarheid - Het systeem moet zo worden geprogrammeerd dat een andere programmeur er makkelijk op kan uitbreiden of 3rd party software aan toevoegen. Zie SDD par. 3.2

## Definities, acroniemen en afkortingen

Er is niets noemenswaardig voorbij gekomen.

## Referenties

Avans BlackBoard

## Overzicht

In de volgende hoofdstukken zal uitgelegd worden welke software architecture wordt gebruikt en welke er voorgesteld word.

# Current Software architecture

Bij dit soort systemen kom je vaak nog een manier van werken tegen van voor het digitale tijdperk, men gebruikt veel papier, gebruikt geen computers en moeten alles archiveren en constant door sturen om anderen op de hoogte te houden.

# Voorgestelde software architecture

## Overzicht

In de volgende paragrafen zal duidelijk worden welke specificaties er zijn gemaakt in het bedenken van het nieuwe systeem. Denk hierbij aan hoe de interne klassen worden verdeeld en motivaties voor het kiezen van bepaalde structuren.

## Subsystem decomposition (UML class diagrams)

Er zal een Model-View-Controller(MVC) patroon worden gebruikt om dit systeem tot stand te brengen.  
Ook is er een server-client communicatie, omdat alle data op een gezamenlijk gebruikte database staat.



In het bovenstaande diagram is te zien hoe de hoofdklasse in de packages communiceren. De database is als package inbegrepen maar wordt mogelijk vervangen door een database communicatie klasse.

MVC houdt in dat de klassen van een bepaalde package grotendeels onafhankelijk van een andere package kan werken. De keuze voor MVC ligt bij de onafhankelijkheid en uitbreidbaarheidsmogelijkheden. Als een programmeur een nieuwe functionaliteit wilt toevoegen aan het autoverhuursysteem zal hij binnen een package kunnen werken zonder dat de andere delen stoppen met werken.  
Voor een kijk in de andere klassen binnen in de packages, zie klassendiagram RAD par. 3.5.4.

### Architectuurstijl

Model handelt de datastructuur en servercommunicatie, view levert het GUI met alle panelen en controller handelt muisklikken en toetsen. De main klasse zal uiteindelijk buiten de packages staan om de hoofdklassen van de packages te initialiseren. De communicatie tussen lagen zal liggen in de hoofdklassen van de packages. Omdat er met Swing componenten in Java wordt gewerkt is er niet expliciet een communicatie met de view package, maar wordt dit intern afgehandeld. De model package zal met de view communiceren via de server in en uitput. Data zal op databases bij de server worden opgeslagen, dus is er geen wrapperklasse nodig om alle data bij elkaar te houden.

### Concurrency

Er zal gebruik gemaakt worden van een client - server/database systeem die tegelijkertijd meerdere clients van filialen moet afhandelen. Hiervoor zal een systeem komen die om de beurt een request van de client afhandelt. De connecties zullen ook op meerdere threads worden afgehandelt. Voor elke client die een connectie maakt, een nieuwe thread.

## Hardware/software mapping (UML deployment diagrams)

De software van de client kan op computers met java gedraaid worden, als de computer in de afgelopen 10 jaar gekocht is zal deze waarschijnlijk goed genoeg zijn.  
Specifiek is er een internetverbinding nodig en een besturingssysteem van windows XP of hoger. Kwa hardware zal een Intel Pentium 4 processor al voldoende zijn met 2GB werkgeheugen. Een hardeschijf van 50gb is ook al voldoende, aangezien de data afgehandeld word op de server.

De server zal iets sterker moeten zijn. Optimaal zou een professionele server alle clients kunnen afhandelen. Op deze server moet ook java komen te staan. Hier komt ook de database dus veel geheugen zal ook van te pas komen.  
Persoonlijk zullen wij aanraden om een HP of Dell server aan te schaffen met één of meerdere dual of quad core processor(en), 2GB werkgeheugen en een harde schijf met 250 of meer GB.

## Persistent data management

Er zal 1 database in het hoofdkantoor staan met de informatie van alle auto's, klanten en filialen. Dit wordt gedistribueerd voor een tijdelijke kopie naar de clients die data aanvragen. Clients worden door de server geverifieerd of ze de data mogen bekijken en stuurt een copy door. De copy wordt bij de client verwijdert op het punt dat het scherm wordt afgesloten of een connectie time-out optreedt.

## Access control en security

Op de server wordt bekeken van welke client de data afkomt. Dit wordt geverifieerd door een usernaam en wachtwoord mee te geven. De volgende gebruikers kennen de volgende rechten:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Gebruiker | Filiaalgegevens | Autogegevens | Klantgegevens | Verkoopgegevens |
| Baliemedewerker | Nee | Ja | Ja | Nee |
| Financiële administrator | Ja | Ja | Nee | Ja |
| Planning medewerker | Ja | Ja | Nee | Nee |
| Server administrator | Ja | Ja | Ja | Ja |

## 

In termen van security zal data die wordt overgestuurd geëncrypt worden met de RSA methode om onderscheppingen te voorkomen. Is er een compromis in het systeem, zal de Server Administrator er een stokje voor steken.

## Global software control

Voor elke afdeling op ene filiaal komt dezelfde client te staan. De bevoegdheden zorgen dat elke medewerker bij zijn eigen schermen blijft. De clients zullen niet onderling communiceren maar dit gaat via de server. Alles wordt in de database gezet en een andere client zal de data hieruit weer opvragen. Er zal één server zijn met de database erop met een serverapplicatie. De serverapplicatie wordt beheerd door alleen de server administrator op het hoofdkantoor.

## Boundary conditions

### Start up

Bij het opstarten zal de gebruiker zich moeten identificeren en vervolgens wordt als default opstarthandeling het overzicht worden getoond van het filiaal.

### shutdown

Bij shutdown zal er worden gekeken of er nog vensters open staan met data erin. De gebruiker krijgt een prompt of de data erin mag worden weggegooit. Zo nee wordt de shutdown gestopt en mag de gebruiker afmaken waar hij nog mee bezig was. Zo ja, zal de applicatie doorgaan met zichzelf afsluiten door de verbinding met de server te verbreken en dan het programma af te sluiten.

### Foutafhandeling

Bij connectieverlies met de server zal er elke 3 seconde gepingd worden om te kijken of de connectie er weer is. Is na 3 keer pingen de verbinding nog niet terug zal de gebruiker een bericht krijgen op het scherm krijgen dat de connectie is verloren en krijgt twee opties. Of hij sluit het programma af en verliest de data of hij laat het systeem doorpingen en proberen de connectie weer op te pakken voor 10 seconden. Na 10 seconden komt dezelfde prompt weer als de connectie niet oppakt.  
Bij kleine fouten (bugs) zal binnen er binnen in de applicatie een help functie zijn. Hierin zit een hardcoded FAQ, en indien het antwoord er niet tussen staat is de email/telefoonnummer van de server administrator daar ook beschikbaar.  
Bij grote fouten, zoals een volledige crash bij de client, zal de informatie niet worden opgeslagen of worden er geen back-ups gemaakt. De applicatie zou moeten worden afgesloten en opnieuw opgestart moeten worden. Omdat data meteen naar de server wordt gestuurd zal dit alleen een kleine irritatie zijn. Voor een crash op de server zal er een back-up worden gemaakt elke dag rond 20:00(Optimaal na werktijden) om schade te beperken. Deze back-ups worden op de server zelf opgeslagen, maar door de opsla directory variabel te maken zal het mogelijk zijn om het op een aparte PC te zetten.

# Subsysteem services

Het systeem bestaat uit 5 packages. De model, view en control van de client en de server met database.  
De model zal communicatie tussen server en client afhandelen en kent de entiteitsklassen van die de database ook heeft. Hiermee kan de client tijdelijk instanties maken van de database informatie.  
De view zal een omvattende window klasse hebben. Hierin zullen aparte tabs beschikbaar zijn voor de diverse invoerschermen die als aparte klassen worden geprogrammeerd. Informatie- of errorschermen worden getoond met standaard Java Swing schermen.  
De controller zal slechts 1 klasse kennen die overerft van de toetsenbord en muis listener. Voor het geval dat bepaalde onderdelen niet door Java Swing worden ondersteund is het mogelijk hier acties mee te bepalen.  
De server kent klassen met betrekking tot het sturen, ontvangen, identificeren, decoderen en encrypten van data. Dit zijn allemaal technische onderdelen die automatisch werken.  
De database kent alle entiteitsklassen net zoals de model package op de client, alleen heeft de database nog een extra wrapper klasse die alle data bevat. Hier worden ook manipulaties op uitgevoerd die via de server binnenkomen.