Inhoudsopgave

[Globale omschrijving Icaras 3](#_Toc423508948)

[Korte omschrijving per module 3](#_Toc423508949)

[Specificaties 4](#_Toc423508950)

[Relatiebeheermodule 6](#_Toc423508951)

[Git-repository 6](#_Toc423508952)

[Configuratie 6](#_Toc423508953)

[Hibernate: 6](#_Toc423508954)

[Spring: 7](#_Toc423508955)

[Apache Tiles: 7](#_Toc423508956)

[Web Configuratie: 7](#_Toc423508957)

[Model 7](#_Toc423508958)

[POJO’s 7](#_Toc423508959)

[Databasemodel 8](#_Toc423508960)

[TODO DAO’s 10](#_Toc423508961)

[TODO Service 10](#_Toc423508962)

[View 10](#_Toc423508963)

[Controller 11](#_Toc423508964)

[TODO Test 11](#_Toc423508965)

[TODO Resources 12](#_Toc423508966)

[TODO Sample data 12](#_Toc423508967)

[TODO Utils 12](#_Toc423508968)

[Hoe te gebruiken (front-end) 12](#_Toc423508969)

[Organisaties / zoeken: 12](#_Toc423508970)

[Organisaties / nieuw: 13](#_Toc423508971)

[Personen / zoeken: 13](#_Toc423508972)

[Personen / nieuw: 13](#_Toc423508973)

[TODO Hoe te gebruiken (back-end) 13](#_Toc423508974)

[Persoon opslaan 13](#_Toc423508975)

[Bedrijf opslaan 13](#_Toc423508976)

[Done 14](#_Toc423508977)

[Work in progress 14](#_Toc423508978)

[To do 14](#_Toc423508979)

[Appendix: 15](#_Toc423508980)

[Termen 15](#_Toc423508981)

# Globale omschrijving Icaras

Icaras is een bedrijfs(-web)applicatie voor RSvier. De naamgeving komt voort uit de verschillende modules waaruit de applicatie wordt opgebouwd: **i**ntake, **c**ursist, **a**rbeidsmarkt, **r**elatie en **a**gendabeheer **s**ysteem.

Het doel van dit document is het vastleggen van de gebruikte specificaties, termen, techniek, structuur, functionaliteit en vervolgstappen. Zodat in de toekomst nieuwe cursisten, stagiaires of beheerders het project zich snel eigen kunnen maken.

Het doel van de applicatie is de bedrijfsprocessen binnen RSvier, qua werving, omscholing en detachering, bij te kunnen houden. De kern van de applicatie is relatiebeheer, cursistvolgsysteem en een arbeidsmarktsysteem gekoppeld aan een database.

De primaire gebruiker is Rob Steutel van RSvier, secundaire gebruikers zijn de Java-docent en beheerder van de applicatie. De user interface bevindt zich in een webbrowser.

## Korte omschrijving per module

**Intake**

Implementeert de functionaliteit met betrekking tot het intakeproces. Aanmeldingen, aanmelders,

status, overeenkomst, afwijzen, accepteren en archiveren.

**Cursisttraject**

Implementeert de functionaliteit met betrekking tot het bijhouden van de voortgang van cursisten:

Welk type traject men volgt, In welke fase men zit, welke boeken in bezit zijn, welke gestuurd moeten worden, welke toets gedaan is met resultaat, en historie van resultaten.

**Arbeidsmarkt**

Implementeert de functionaliteit voor het bijhouden van arbeidsbemiddeling voor inzet van

kandidaten bij bedrijven. Hieronder vallen aanbiedingen die worden gedaan bij bedrijven, het

toewijzen, afwijzen, welke redenen voor afwijzen, contractgegevens, urenspecificatie, declaraties,

andere afspraken en facturering.

**Relatiebeheer**

De module voor het toevoegen, wijzigen, verwijderen, zoeken en ophalen van de relaties van RSvier uit de database.

**Agendabeheer**

Implementeert de functionaliteit voor het automatiseren en synchroniseren van afspraken ingevoerd

in Icaras of smartphone.

## Specificaties

Wat moet de gebruiker met de applicatie kunnen doen.

Algemeen:

* Inlogfunctionaliteit
* Authenticatie gebruikers

Primaire gebruiker:

* Zoeken op naam van bedrijf
* Zoeken op plaats
* Zoeken op naam persoon
* Zoeken op expertise in kandidaten
* Toevoegen van relaties
* Verwijderen van relaties
* Wijzigen van relaties
* Ophalen van gegevens
* Printen van gegevens
* Uploaden van CV’s

Secundaire gebruikers:

Docent:

* Invoeren van gegevens
* Ophalen van gegevens
* Wijzigen van gegevens
* Printen van gegevens

Cursist:

* Inzien resultaten-gegevens
* Wijzigen eigen adres

Beheerder:

* Onderhouden van de pagina’s en database (lookup-tables)

# Relatiebeheermodule

De module die de functionaliteit implementeerd om relaties van RSvier te beheren. Deze relaties zijn opgedeeld in bedrijven en personen. Een persoon kan één of meerdere persoonsrollen hebben wat aangeeft wat voor type relatie desbetreffend persoon heeft met RSvier. Deze rollen zijn gedefiniëerd als cursist, kandidaat, werknemer, stagiair, contactpersoon en privé.

-Een cursist is een persoon die het omscholingstraject bij RSvier volgt.

-Een kandidaat is een persoon die klaar of bijna klaar is met het omscholingstraject en gedetacheerd kan worden bij een bedrijf.

-Een werknemer kan een werknemer(bijv. docent) zijn bij RSvier of iemand die vanuit RSvier gedetacheerd is bij een bedrijf.

-Een stagiair is een persoon die stage loopt bij RSvier.

-Een contactpersoon is een persoon die fungeert als vertegenwoordiger en/of contactpunt is bij een bedrijf.

-Privé geeft aan dat een persoon een persoonlijke relatie is van Rob Steutel van RSvier.

Een beperkte combinatie van rollen is mogelijk, zo kan iemand zowel stage lopen bij RSvier en klaar zijn om gedetacheerd te worden.

De relatiebeheermodule is opgebouwd uit meerdere abstractie-lagen conform het MVC(Model view controller)-design pattern. In de komende paragrafen wordt uitgelegd welk databasemodel er gebruikt is, welke abstractielagen er zijn, welke klasses er geïmplementeerd zijn, waar deze te vinden zijn en hoe je de relatiebeheermodule gebruikt.

## Git-repository

Het project staat op de Gitlab repository van RSvier: <http://77.61.99.106:10080/icaras/relatie-beheer.git>. Tijdens de ontwikkeling van dit project is gebruikt gemaakt van een Git-client genaamd SourceTree, van Atlassian (<https://www.sourcetreeapp.com/>).

## Configuratie

### Hibernate:

/Icaras/src/main/resources/icarasdb-context.xml => configuratiefile voor Hibernate. De waardes staan in icarasdb.properties

/Icaras/src/main/resources/icarasdb.properties => instellingen voor de database connectie. Op dit moment wordt er nog gebruik gemaakt van een locale database, dus de instellingen zal per persoon verschillen.

jdbc.driver=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql://localhost:3306/icaras

jdbc.username=

jdbc.password=

hibernate.dialect=org.hibernate.dialect.MySQLDialect

hibernate.hbm2ddl=update

hibernate.showsql=true

hibernate.lazy\_load=true

### Spring:

/Icaras/src/main/webapp/WEB-INF/icaras-servlet.xml

annotation driven

component-scan base-package => zoekt controller classes op basis van @Controller annotatie in nl.rsvier.icaras.controller

ipv een standaard InternalResourceResolver, gebruikt dit project een TilesViewResolver ivm Apache Tiles

### Apache Tiles:

/Icaras/src/main/webapp/WEB-INF/icaras-servlet.xml

tilesViewResolver => default layout staat in /WEB-INF/layouts/defaults.xml

### Web Configuratie:

/Icaras/src/main/webapp/WEB-INF/web.xml

contextConfigLocation verwijst naar icaras-context.xml

ContextLoaderListener

RequestContextListener

DispatcherServlet: /Icaras

## Model

### POJO’s

Hieronder vind je de POJO’s die gebruikt worden in het kader van relatiebeheer. Sommige zijn vanzelfsprekend, anderen dienen wat uitleg. De POJO’s zijn te vinden in **src/main/java/nl/rsvier/icaras/core/relatiebeheer**.

**Persoon**: alle menselijke relaties van RSvier worden met behulp van deze klasse opgeslagen. Een persoon heeft verplicht (niet-nullable) een voornaam, achternaam en een geslacht. Een persoon kan meerdere identiteitsbewijzen, adressen, digitale adressen en persoonsrollen hebben.

**Persoonsrol**: geeft de rol aan van een persoon. Persoonsrol is de tussen-entiteit die een rol verbind aan een persoon. Een persoonsrol heeft verplicht (niet-nullable) een begindatum, persoon en een rol. Door de begindatum op te nemen en een eventuele einddatum kan, bijvoorbeeld, bijgehouden worden wanneer een persoon gestart is bij RSvier als cursist, en wanneer hij/zij klaar is.

**Rol**: type rol die gelinked kan worden aan een persoon via persoonsrol. Bij het ontwikkelen van de relatiebeheermodule zijn de hierna genoemde rollen gedefinieerd: cursist, kandidaat, werknemer, stagiair, prive en contactpersoon.

**Identiteitsbewijs**: een identiteitsbewijs van een persoon. Deze entiteit heeft geen auto-generated Id, maar het documentnummer is het Id. Een identiteitsbewijs heeft verplicht een identiteitsbewijs-type, persoon en vervaldatum.

**IdentiteitsbewijsType**: type identiteitsbewijs die gelinked kan worden aan een persoon via identiteitsbewijs.

**Adres**: een adres van of een persoon of een bedrijf. Een adres heeft verplicht (niet-nullable) een straatnaam, nummer, postcode, plaats en land.

**AdresType**: type adres die gelinked kan worden aan een persoon of bedrijf via adres.

**DigitaalAdres**: onder digitaal adres vallen o.a. telefoonnummer, e-mail, linkedin etc. van een bedrijf of persoon. Een digitaal adres heeft verplicht (niet-nullable) een omschrijving, contactvoorkeur(boolean) en een digitaal adres-type.

**DigitaalAdresType**: type digitaal adres die gelinked kan worden aan een persoon of bedrijf via digitaal adres.

**Bedrijf**: een bedrijf waar RSvier een relatie mee heeft. Een bedrijf kan contactpersonen(type rol) waarmee RSvier contact mee heeft.

**BedrijfType:**  geeft het type aan van een bedrijf. Bijvoorbeeld leverancier, detacheerder, eindklant.

**BedrijfExpertise:** geeft de expertises (of specialisaties) aan van een bedrijf.

**Expertise:** Bijvoorbeeld Java, PHP, .net, spring, hibernate etc.

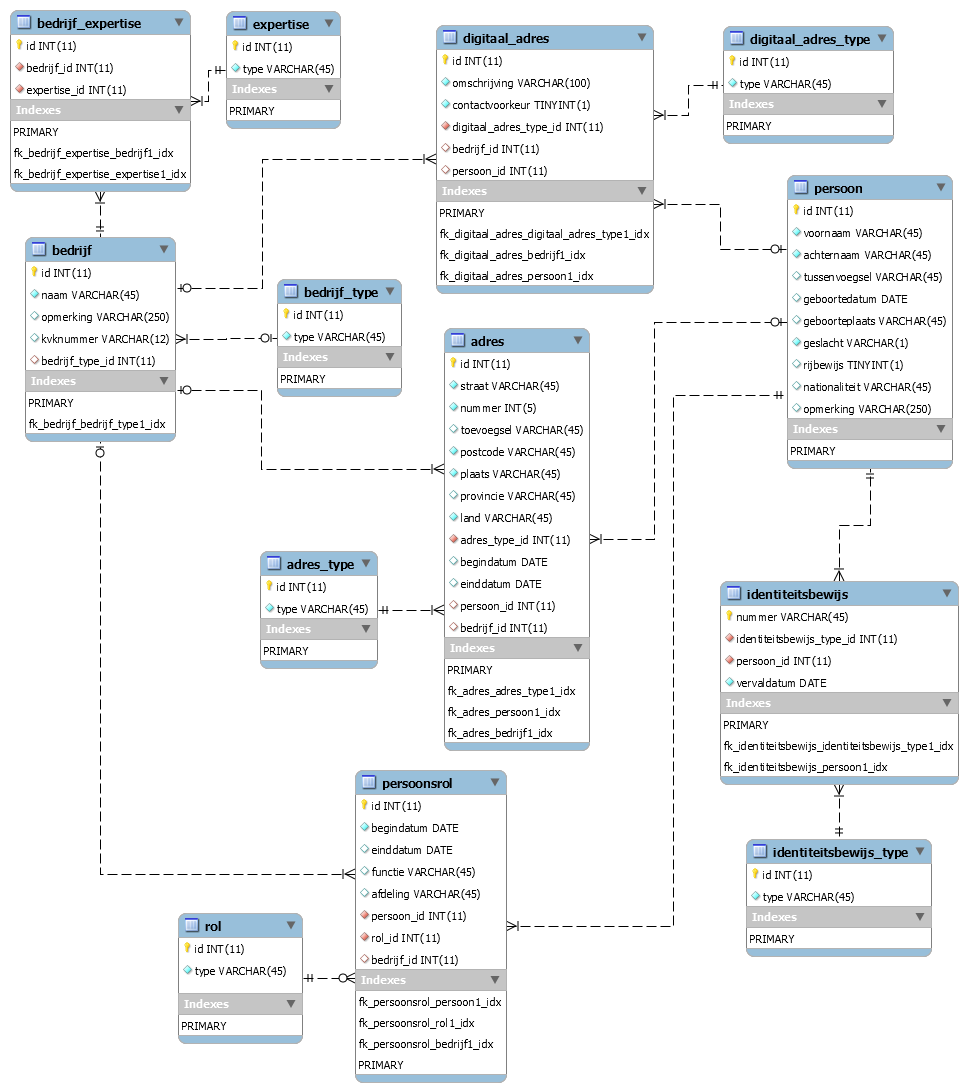
In elke POJO zijn de equals() en hashCode() methode overriden. In deze methodes wordt het id, wat automatisch gegenereerd wordt bij het opslaan in de database, en de relaties naar andere entiteiten niet meegenomen in de vergelijking. Dit is zo gedaan om ervoor te zorgen dat objecten uit de database (met id en relaties) alsnog vergeleken kunnen worden met objecten die nog gesaved moeten worden.

Voor uitleg van de overige methodes in de POJO’s zie de javadoc in betreffende klasses.

### Databasemodel

Het databasemodel is op te bouwen via hibernate in de applicatie of in de database zelf. In deze applicatie is er gekozen voor het laatste om er zeker van zijn dat het model klopt. Daarnaast is het mogelijk om dit model te gebruiken voor applicaties in andere programmeertalen.

Het databasemodel is ontwikkeld in MySQL workbench. Het MySQL workbench model is te vinden in de **project\_documenten folder** en **Figuur1.** Het model spreekt in principe voor zich, voor uitleg van de entiteiten zie voorgaande paragraaf over **POJO’s**. Voor de uitbreidbaarheid, onderhoudbaarheid en het reduceren van redundantie van de applicatie is er gekozen voor het gebruik van lookup-tables. Deze tabellen zijn de entiteiten **rol, digitaal\_adres\_type, adres\_type, identiteitsbewijs\_type, expertise en bedrijf\_type**. Mocht er een nieuw type, expertise of rol bijkomen, dan is het mogelijk om deze toe te voegen aan de lookup-table in de database. Er is dus geen nieuwe klasse hiervoor nodig.



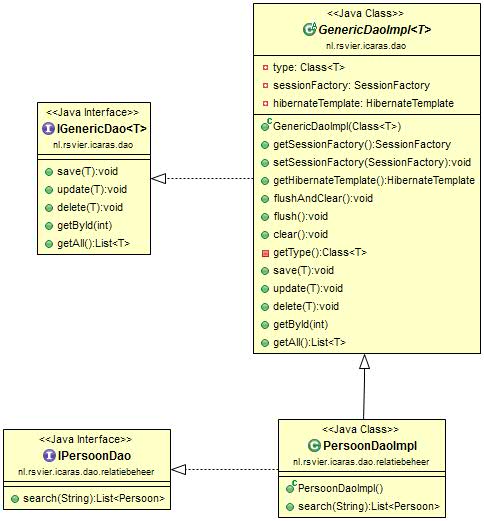
**Figuur 1**: **Databasemodel icaras-relatiebeheer.** Illustreert de verschillende entiteiten in relatiebeheer compleet met attributen en relaties onderling.

#### Scripts

De SQL scripts om de lookup-tables te vullen en de tabellen te legen zijn te vinden in de **project\_documenten folder**. Deze scripts kunnen in de MySQL workbench uitgevoerd worden.

### DAO-laag

De DAO-laag is gesplitst in twee delen, generieke DAO en de DAO implementatie. De generieke DAO zijn te vinden in **src/main/java/nl/rsvier/icaras/dao/** en de implementatie voor deze module in **src/main/java/nl/rsvier/icaras/dao/relatiebeheer**. Hieronder in **figuur 2,** een voorbeeld van een implementatie.



**Figuur 2: DAO-laag.** Voorbeeld van een implementatie van de generieke DAO, in dit geval voor Persoon.

De DAO-laag is opgebouwd uit een **IGenericDao<T>** interface, **GenericDaoImpl<T>** klasse, **“POJOnaam”DaoImpl klasse** en een **I”POJOnaam”Dao** interface.

-De **IGenericDao<T>** interface specificeert alle (algemene) CRUD-methoden die elke DAO moet implementeren.

-Een generieke implementatie van de CRUD-methoden is gegeven in de **GenericDaoImpl<T>**.

-De **I”POJOnaam”Dao** interface specificeert alle specifieke methoden die geïmplementeerd worden in **“POJOnaam”DaoImpl**.

-**“POJOnaam”DaoImpl** erft de algemene CRUD-methoden van **GenericDaoImpl<T>**. In de constructor wordt de klasse meegegeven aan de (super-)constructor van de generieke DAO.

**PersoonDaoImpl** als voorbeeld:

public PersoonDaoImpl() {

super(Persoon.class);

}

Constructor in **GenericDaoImpl<T>:**

public GenericDaoImpl(Class<T> type) {

this.type = type;

}

Tevens implementeert de **“POJOnaam”DaoImpl** specifieke methoden gespecificeert in **I”POJOnaam”Dao**. “POJOnaam”DaoImpl is geannoteerd met @Repository(“naam”) voor de Spring automatic bean detection via component-scan.

### TODO Service-laag

De service-klasses in icaras-relatiebeheer

* AdresService
* BedrijfService
* DigitaalAdresService
* IdentiteitsbewijsService
* PersoonService
* PersoonsrolService

## View

**Apache Tiles** – dit Framework werkt combinatie met Spring MVC. Hiermee is het simpel om een pagina op te delen in delen (tiles). Ook voorkomt het dat je steeds code moet herhalen (denk aan header/footer en menu).

/Icaras/src/main/webapp/WEB-INF/templates/default.jsp: dit is de standaard pagina, hierin staan de verwijzing naar de css file en de imports voor de libraries. (tablibs moeten wel per tile nog geimporteerd worden)

De layout van de pagina wordt bepaald door /Icaras/src/main/webapp/resources/rs4/css/icaras.css

/Icaras/src/main/webapp/WEB-INF/layouts/default.xml: hierin staat de configuratie van alle pagina's.

home.base is de basisconfiguratie. Nieuwe pagina's extenden home.base of eventueel een andere pagina. De bestaande tiles staan in /Icaras/src/main/webapp/WEB-INF/tiles en submappen.

**Spring form:** De forms zijn gekoppeld aan een Object, dat een modelAttribute is. In de Post Controllermethode, creeert Spring een nieuw Object. Alle attributen van het originele Object moeten daarom in de form worden meegegeven, eventueel als hidden field.

**Data Transfer Objects:** op sommige pagina's wordt gebruik gemaakt van DTO's. Voor meer informatie zie Controller.

**jQuery:** wordt op dit moment gebruikt om de rows clickable te maken. In de toekomst zal jQuery ook worden gebruikt voor onder andere form validatie en tabellen sorteren

**Tomcat:** Het project is getest op een Tomcat v7.0 Server

## Controller

De controllers in icaras-relatiebeheer:

* HomeController
* ContactpersonenlijstController
* OrganisatielijstController
* PersonenlijstController
* RelatiebeheerController

Er is geprobeerd een logische opdeling te vinden voor de controllers. Op dit moment gebruiken we voornamelijk PersonenlijstController en OrganisatielijstController. Dit zal in de toekomst nog aangepast worden.

**Data Transfer Objects:** deze staan in /Icaras/src/test/java/nl/rsvier/icaras/util/relatiebeheer. De DTO's geven meer vrijheid en mogelijkheden in een form.

ZoekInput: bevat de zoekString + begin en eindtijd van een zoekopdracht

**BedrijfDTO:** voornamelijk om meerdere objecten in 1 form te zetten. Een bedrijf heeft bv geen verwijzing naar een Persoon, maar een List van Persoonsrollen, die een Persoon bevatten.

**PersoonDTO:** wordt gebruikt voor het toevoegen van een nieuw persoon, 2e adres en extra persoonsrol.

Personenlijst wordt gesorteerd op achternaam met de AchternaamComparator. Organisatieslijst wordt gesorteerd op Bedrijfsnaam en daarna plaatsnaam met de BedrijfComparator.

## TODO Test

#### POJO-test

De POJO-tests in icaras-relatiebeheer:

* AdresTest
* BedrijfTest
* CoreTestSuite
* DigitaalAdresTest
* IdentiteitsbewijsTest
* PersoonsrolTest
* PersoonTest

#### DAO-test

De DAO-tests in icaras-relatiebeheer:

* AdresDaoImplTest
* BedrijfDaoImplTest
* DaoTestSuite
* DigitaalAdresDaoImplTest
* IdentiteitsbewijsDaoImplTest
* PersoonDaoImplTest
* PersoonsrolDaoImplTest

## TODO Resources

De resources in icaras-relatiebeheer:

## TODO Sample data

## TODO Utils

* BedrijfDTO
* PersoonDTO
* Zoekinput

## Hoe te gebruiken (front-end)

### Organisaties / zoeken:

**Zoekpagina:**

zoeken kan op basis van een bedrijfsnaam en/of plaats

de naam en/of plaats mogen uit meerdere woorden bestaan

zoekt ook op het begindeel/letter van een naam en/of plaats

kan eventueel ook op een einddeel zoeken door middel van een % wildcard (bv %terdam)

**Resultatenlijst:**

de regels zijn klikbaar en leidt tot de detailspagina

sorteren op naam of plaats moet nog worden geimplementeerd

op elke regel staat nu een link 'verwijder', dit kan eventueel worden vervangen door een checkbox om meerdere regels te selecteren en tegelijk te verwijderen (dit is afhankelijk van wat de opdrachtgever wil)

verwijder leidt naar bevestigpagina

**Detailspagina:**

naam en opmerking

1 of meer adressen (bezoekadres is meestvoorkomend en dus standaard)

AdresTypeId staat hardcoded op dit moment nog hardcoded in de pagina

rechts staan de contactpersoongegevens

contactvoorkeur staat nu nog als radiobutton, kan eventueel vervangen worden door vinkjes of checkboxes

alle velden zijn te veranderen (klik Update of druk op Enter om de verandering te bevestigen

**Bevestigingpagina:**

extra adres(sen) toevoegen (opent adresform, leidt daarna terug naar de bevestigpagina

nieuw contactpersoon (opent nieuw form, leidt daarna terug naar de bevestigpagina)

bestaand persoon toevoegen als contactpersoon (opent zoekscherm met personenlijst)

regels in de personenlijst zijn klikbaar en voegt deze persoon toe als contactpersoon voor dit bedrijf (leidt daarna terug naar de bevestigpagina)

### Organisaties / nieuw:

**Nieuwe organisatie:**

naam en eventueel opmerking

standaar bezoekadres

**Bevestigingpagina:**

dezelfde pagina als bij organisatiezoeken, met dezelfde opties (zie boven)

### Personen / zoeken:

**Zoekpagina:**

zoeken kan op basis voornaam en/of achternaam, tussenvoegsel met achternaam, voornaam met tussenvoegsel en achternaam

namen mogen uit meerderen delen bestaan (zoek String wordt gesplitst op spatie en

zoekt ook op het begindeel/letter van een naam

kan eventueel ook op een einddeel zoeken door middel van een % wildcard (bv %an)

**Personenlijst:**

-voornaam, achternaam, tussenvoegsel en rollen

-rows clickable (todo)

-sorteren op kolom (todo)

**Persoondetails:**

-persoonsgegevens links, adresgegevens midden en rollen rechts (scherm moet nog beter worden gebruikt/opgedeeld (todo)

-update van gegevens (todo)

### Personen / nieuw:

**Nieuw persoon:**

persoongegevens + 1 adres standaard

toevoegen leidt naar bevestigingspagina

**Bevestigingspagina:**

kan extra adres toevoegen (leidt weer naar bevestigingspagina)

kan persoonsrol toevoegen (leidt naar placeholder pagina)

## TODO Hoe te gebruiken (back-end)

### Persoon opslaan

### Bedrijf opslaan

## Done

Het model staat in principe vast. Het ERDmodel wordt gebruikt voor een locale database en staat nog niet op de Icaras server bij RSVier. De POJOs komen overeen met het model.

De DAOs en bijbehorende JUnit tests zijn grotendeels af. Afhankelijk van de gewenste functionaliteit aan de voorkant, kunnen er altijd nog extra methodes aan worden toegevoegd (bv search). De Service classes zijn op dit moment concrete klassen, conventie is coding to interfaces, daarom moet er in de toekomst waarschijnlijk nog een abstractie laag tussen.

## Work in progress

Het idee achter de voorkant is dat je vanaf de startpagina's kan kiezen uit de verschillende modules. Op dit moment is alleen Relatiebeheer werkend. In het submenu is nu Personen en Organisaties werkend. Het is nu al mogelijk om nieuwe personen en organisaties toe te voegen. Ook is het mogelijk om zowel nieuwe als bestaande personen aan een organisatie toe te voegen als contact persoon.

## To do

-interfaces in de Service laag

-user profiles en login

-jQuery form validation

-datum input en output format

-autocomplete adresform met postcode en huisnummer

-maillijst met meerdere zoekfilters

-printfunctie voor adressenlijst (kerstkaartenactie)

## Appendix:

### Termen

Relatie

Cursist

Kandidaat

Werknemer

Contactpersoon

Stagiair

Bedrijf

Adres

Adrestype

Digitaal-adres

Digitaal-adres-type