Kieler Kaffee Klub K^3 Project[*]

Witzany, Jan   Luick, Bastian   Boottawong, Juti

2019
July

Abstract

Dies ist eine kurze Zusammenfassung der Inhalte des in deutscher Sprache verfassten Dokuments.

[*]No procrastination

Contents

WORKING TITLE::KAFFEESATT

Bastian Luick(1018266), Jan Witzany(1011713),
Juti Boottawong(1030476).

Scope and Specifications of the Project

To provide miscellaneous information about coffee localities through a web application with students, inbound tourists and coffee fanatics in Kiel.

VISION

Our Vision is that everyone know where they can find their suitable beverage place.

MISSION

Provide a sophisticated web application for students, inbound tourists or coffee fanatics to discover a place to relax and enjoy their favorite coffee and supply themselves with coffee making utensil.

W*H

Who will be using the system?
Students, coffee fanatics and inbound tourists that are in Kiel.
When will be the system be used?
Breaks and Lunches.
Where is the information system used?
Desktop and Mobile at home, at work, on the go, in the city, near sights.
What is represented in the system?
Available coffee sorts, price-range, picture gallery, ratings from *Google* etc., direct links to places, misc. information about coffee (fair trade, preparation process, quality criteria, provenance etc..)
How will the system be used?
Desktop and Mobile via web browser.
Why is the system used?
To find the place to enjoy coffee or buy coffee accessories.
What is the policy, intention, goal, and aim of the provider?
To share our love and knowledge of coffee with coffee drinkers, coffee providers and coffee makers in Kiel.

User & Scenarios Outline

Students much free time, high mobility in the area of Kiel (Student Ticket), bicycle routes, price sensitive
Student thirst for coffee at the university during lectures.

Inbound Tourist no knowledge of Kiel, high price tolerance.

Tourist is in the middle of the city during a day trip and want to relax with a coffee.

Coffee fanatics want to know everything about the coffee or the coffee supplements high expectation, high demand of information,
At home and want to explore novel coffee localities.

1 System Requirements

Speed

- Navigation $<$ 3 second response time.
- Filtering shops and equipment categories $<$ 5 second response time

Product Environment

- The client must be connected with the internet during use of application
- Application works only on the following browsers Firefox, Chrome, Edge, Safari.

Privacy Policy

- Delete permanently on request user account and his reviews.
- Must allowed cookie to locate the user location.
- Password are encrypted.

Localizability

- User interface components are in mostly german or rather in language which is used by the younger people of germany.
- Accept german specific language as input.

2 Requirement

2.1 User Requirements

2.1.1 User Types

Possible properties of every user type: vegan, possibility to use own mug, reusable mug, can speak and read German, every user can paid by cash, age 16-35.
Roles {Content Provider(Admin, ContentManager), User (All user types)}

Admin
Preferences: List of registered user and List of content

Behavior: Interact through desktop with web application for various task
Constraints:
Demands: Access to all content and user information possibility to delete user and add content manager.
Tasks: {CRUD of all content and user account}
Content Manager
Preferences: Concrete and specific input options.(saved options)
Behavior: Want to upload a bulk of content and previews his inputs. Watch out for changes in the coffee shops.
Constraints: must have preview of create or edit content
Demands: Input pages for various content.
Tasks: {CRUD content}
Students
Preferences: Cheap coffee, place with wlan, near bus station, buy with bitcoin
Behavior: User for orientation mobile devices and get to the locations mainly with bicycle or bus
Constraints: low funds, short on time.
Demands: Student wants to drink coffee and possibly a place to work.
Tasks: {filtering, search, look up, navigation, delete own profile, change own mail, rate shops}
Coffee fanatics
Preferences: High quality coffee, parking lot, wlan, preferable possibility to see coffee making process, have a list of favorites
Behavior: User desktop and mobile devices to find misc. information about coffee shops in Kiel. Is content with paying more than average coffee price for high quality coffee.
Always looking for new shops and coffee beverages.
Constraints: No big companies or franchises.
Demands: Fanatics to experience novel coffee specialities in kiel and buy coffee making utensils.

Tourists
Preferences: Nearby current location, card payment
Behaviour: Use mobile devices to find coffee shops in Kiel to relax and drink coffee. Is usually near sights.
Constraints: Low mobility, doesn't know localities, short on time, no big companies or franchises.
Demands: Local cafe shops that are nearby

2.1.2 User Stories

Table 1: User Story: User filtering options

User Story ID:	1		
User Story Name:	Search coffee place through filtering		
Created by:	KKK	Date created: July 4, 2019	
Roles	Students Coffee fanatic Tourists		
Description:	The User is on the website and use the presented filtering options to look up shops.		
Preconditions:	1. Know what filtering options mean.		
Postconditions:	Is presented list of shops		
Trigger:	Search button		
Flow:	1. Click on available filtering options 2. filtering results are showed 3. browse through list		

Table 2: User story detail

User Story ID:	2		
User Story Name:	User quick search		
Created by:	KKK	Date created: July 4, 2019	

Table 2 – *Continued on next page*

Table 2 – *Continued from previous page*

Roles	Student Tourist
Description:	User is on a break and are looking for a nearby coffee shop and use quick search function
Preconditions:	1. Is on our landing page 2. Click quick search button
Postconditions:	Get a list of nearby coffee shops
Trigger:	Search button
Normal flow:	1. the user clicked on the search button;

Table 3: User Story Evaluation

User Story ID:	3		
User Story Name:	Evaluate coffee shop		
Created by:	KKK	Date created: July 4, 2019	
Roles	Student Coffee fanatic		
Description:	Evaluate Coffee shops and write a review		
Preconditions:	1. The user is logged in.		
Postconditions:	Can see his evaluation about the shop.		
Trigger:	Star symbol		
Normal flow:	1. Click on a specific shop. 2. Click on star symbol.		

Table 4: User Story: Search equipment

User Story ID:	4

Table 4 – *Continued on next page*

Table 4 – *Continued from previous page*

User Story Name:	Search equipment		
Created by:	KKK	Date created: July 4, 2019	
Roles	Coffee fanatic		
Description:	Is on the equipment subpage and select filtering options.		
Preconditions:	1. Is on the equipment subpage		
Postconditions:	Show shops that fits the selected filtering options.		
Trigger:	Filtering options		
Flow:	1. Select filtering options		

Table 5: User story detail

User Story ID:	5		
User Story Name:	Edit review		
Created by:	KKK	Date created: July 4, 2019	
Roles	All		
Description:	User edit reviews		
Preconditions:	1. The user is logged in 2. He has given reviews		
Postconditions:	Review was edited		
Trigger:	Star symbol on the shop page		
Normal flow:	1. User clicked on star symbol.		

Table 6: User Story Manage Content

User Story ID:	6
User Story Name:	Manage Content

Table 6 – *Continued on next page*

Table 6 – *Continued from previous page*

Created by:	KKK	Date created: July 4, 2019	
Roles	Content-Manager Admin		
Description:	The Actor can add, edit or remove content {shop, equipment, informations, events} (do **C**reate**R**ead**U**pdate**D**elete operations on content)		
Preconditions:	1. The user is logged in. 2. Is on the input page.		
Postconditions:	Selected CRUD executed on database		
Trigger:	CRUD button		
Flow:	1. Fill the input forms. 2. Click either on save, delete or publish.		

Table 7: User story detail

User Story ID:	7		
User Story Name:	Login		
Created by:	KKK	Date created: July 4, 2019	
Roles	Student Coffee fanatic Tourist Content-Manager Admin		
Description:	The user give in the account information and presses the login button.		
Preconditions:	1. user is not logged in 2. user is registered 3. account data is correct		

Table 7 – *Continued on next page*

Table 7 – *Continued from previous page*

Postconditions:	User is logged in the system and is redirect to account last page. And has access to the corresponding functionality.
Trigger:	Login button
Normal flow:	1. Give account data 2. click on login

Table 8: User story detail

User Story ID:	8		
User Story Name:	Registration		
Created by:	KKK	Date created: July 4, 2019	
Roles	Student Tourist Coffee fanatic		
Description:	User register on the website.		
Preconditions:	1. Is on the registration page. 2. Fill out formula with correct data		
Postconditions:	Is registered		
Trigger:	Register button		
Flow:	1. Fill out input forms 2. system checked input 3. click on registration button		

Table 9: User story detail

User Story ID:	9		
User Story Name:	Delete account		
Created by:	KKK	Date created: July 4, 2019	

Table 9 – *Continued on next page*

Table 9 – *Continued from previous page*

Roles	User Admin
Description:	User delete account
Preconditions:	1. User has account 2. User is logged in
Postconditions:	Is automatic logged out of application and all reviews of the user are deleted.
Trigger:	Delete button
Flow:	1. User clicked on delete button. 2. Verify in popup his deletion request. 3. Click delete button.

Table 10: User story detail

User Story ID:	10		
User Story Name:	Manage registered user		
Created by:	KKK	Date created: July 4, 2019	
Roles	Admin		
Description:	Admin create content-manager and delete every other account		
Preconditions:	1. User has account 2. User is logged in		
Postconditions:	DELETE operation on database on selected user account and his reviews.		
Trigger:	Button corresponding to the action		

Table 10 – *Continued on next page*

Table 10 – *Continued from previous page*

Flow:	1. Select user account 2. Click delete button 3. Popup 4. Select yes

Table 11: User Story: Logout

User Story ID:	11		
User Story Name:	Logout		
Created by:	KKK	Date created: July 4, 2019	
Roles	All		
Description:	The User is on the website and use the logout button		
Preconditions:	1. User is logged in		
Postconditions:	Is logout		
Trigger:	Logout button		
Flow:	1. user clicked on the logout button.		

3 Mini-Stories

Search coffee shop through filtering(all)
Landing page, search page
Preconditions: free access
Actions: select preferences to filter the shops
PostCond: shop list is updated corresponding to the selected options

Evaluate coffee shop first time(registered user
Every page
Preconditions: Free access
Actions: Skip to login, log as corresponding role
Postconditions: Is logged as user, stayed on shop site

Content[shop]

Preconditions: (Logged in)
Action: (Evaluate shop)
Postconditions: (Evaluation of user saved to user account and recalculate average rating)

Manage Content (admin, content manager)
Every page
PreCondition:Free access
Actions: Skip to login, log as corresponding role
PostCond: Is logged in, is on account management

Account Management
Preconditions(logged in)
Actions: (Skip to Content Management)
Postconditions: (Is on content management)

Content Management
Preconditions: (Logged in)
Actions: (CRUD action on content)
Postcondition: (Corresponding crud action on content)

4 SiteLang Specification

The following figures show various and distinct flow, structure and behaviour of the web information system from KAFFEESATT web application. Specifications: On every page there is the navigation bar. Furthermore it is possible to login or logout on every page as well. If user is not log in and want to use a log in feature he will be directed to the login input form.

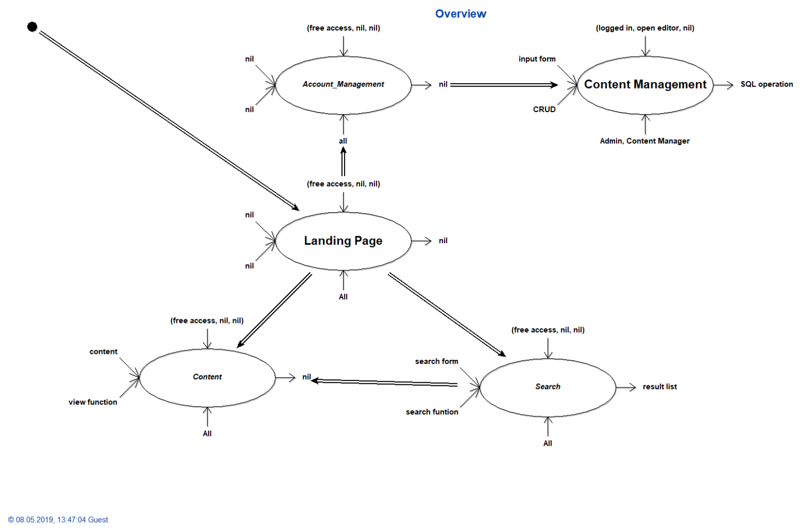

Figure 1: Overview of KAFFEESATT

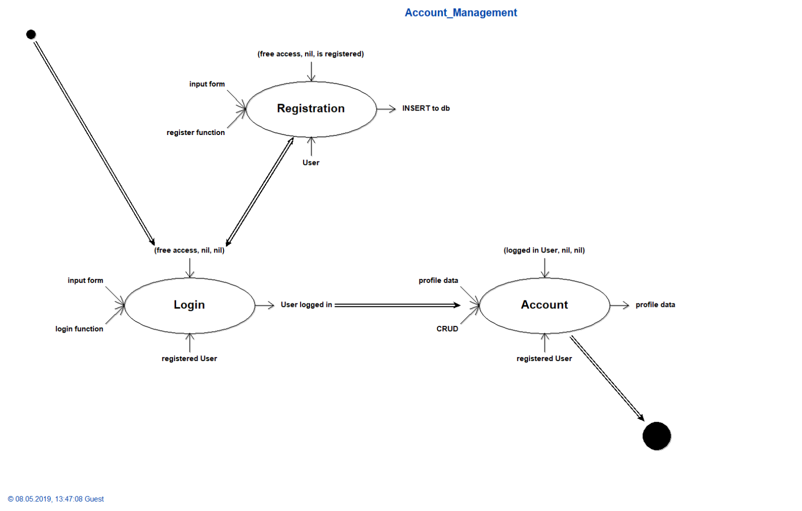

Figure 2: Account Management of KAFFEESATT

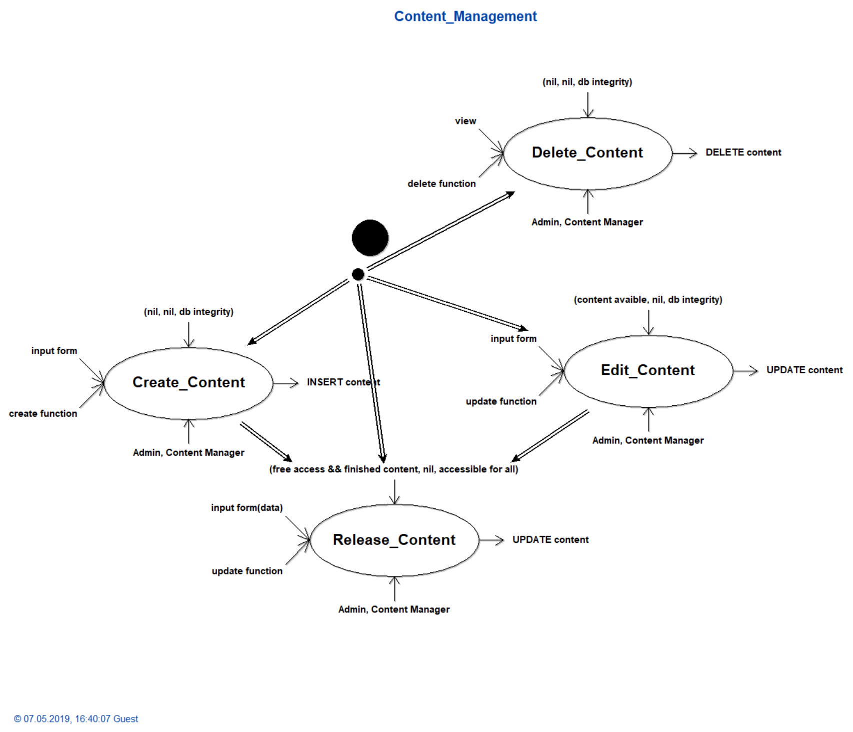

Figure 3: Content Management of KAFFEESATT

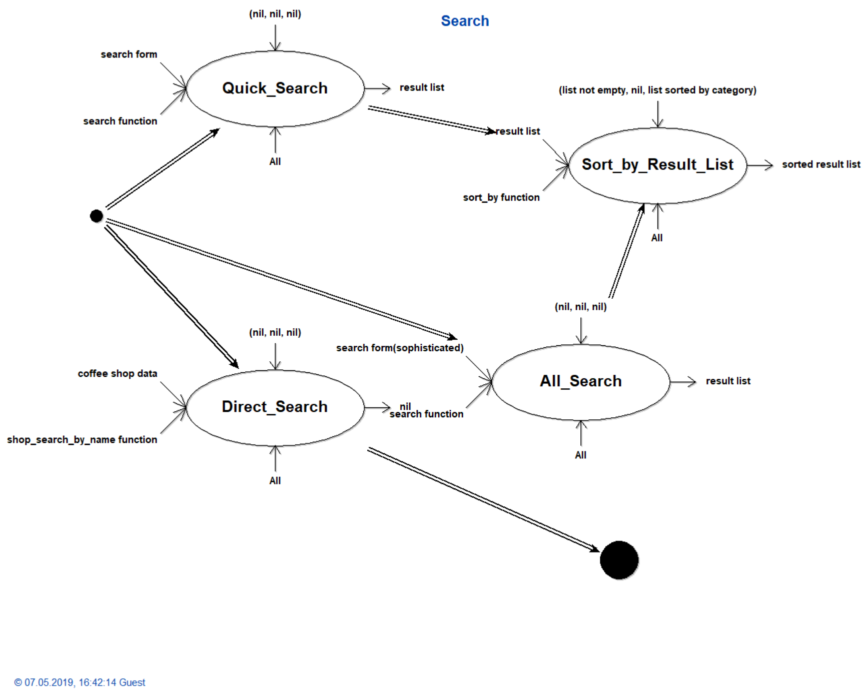

Figure 4: Search of KAFFEESATT

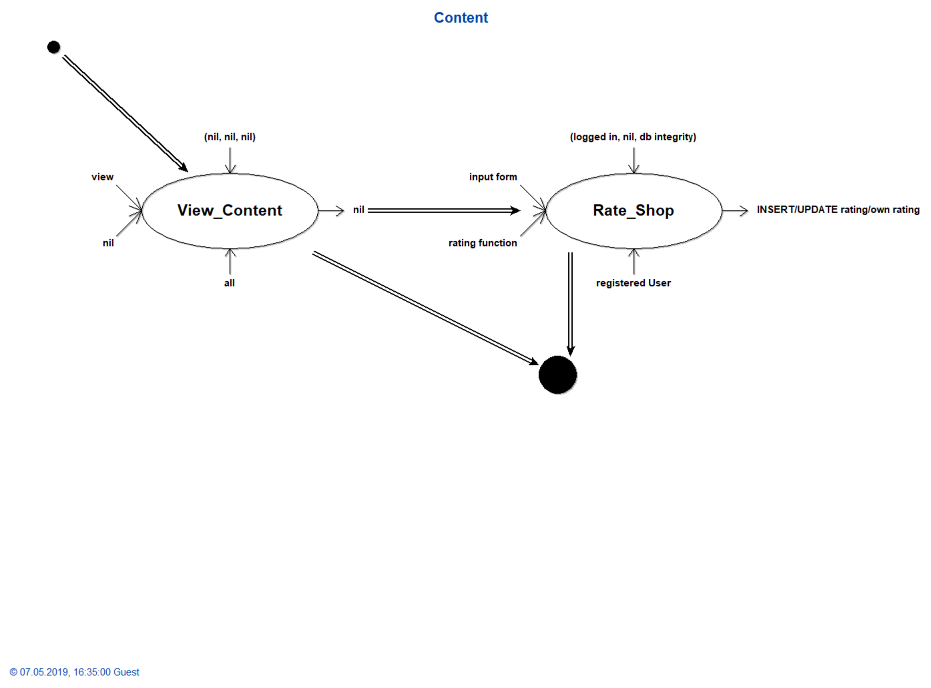

Figure 5: Content KAFFEESATT

4.1 SiteLang Functionality by Scene

Defintions

SETs

- Content are the following entities C := {Shop} with their following attributes.
- Article are the following entities A := {Blend, Beans, Coffee_Drink, Equipment}.
- User are the following entity U := {User and their specialization}.

FUNCTIONs

- $filter :: (C \times filterContent) \rightarrow Boolean : x \mapsto$ if Content satisfied filter flags: return true; else false;
- $filter :: (A \times filterArticle) \rightarrow Boolean : x \mapsto$ if Article satisfied filter flags: return true; else false;
- $filterContent :: C \rightarrow Value : \{C.Attributes\}$ ={poi, workstation, equipment, wlan, outdoor, fair_trade, child_friendly, disabled_friendly, latte_-art, pet_friendly, food, franchise, price_class }
- $filterArticle :: A \rightarrow Value : \{A.Attributes\}$ ={category, sub_category}
- $reduced(filterContent) :: \{quickserch(X) | X \in C.Attributes\}$ ={POI, Workstation, Rösterei }
- $id : (C \cup A) - > id : x - >$ give the primary key of x
- $Result - List(X)$: List of members of Set X
- $Result(X)$: specific member of Set X

Functionality by Scence

Overview
Scene (Content-Management)
View (in) Input-Form(C || A)
View (out) Execute corresponding SQL command

Scene (Search)
View (in) Input-Form(C)
View (out) Result-List(C)

Scene (Content-Management)
View (in) Input-Form(C)
View (out) INSERT/READ/UPDATE/DELETE(C)

Scene (Content)
View (in) Content

ContentManagment
Scene (Create_Content)
View (in) Input-Form(C || A)
View (out) INSERT(C || A)

Scene (Release_Content)
View (in) Input-Form(C || A)
View (out) UPDATE(C || A)

Scene (Edit_Content)
View (in) Input-Form(C || A)
View (out) UPDATE(C || A)

Scene (Delete_Content)
View (in) View(C || A)
View (out) Delete(C || A)

Content
Scene (View_Content)
View (in) View(C || A)

Scene (Rate_Shop)
View (in) Input-Form(C.Rating)
View (out) INSERT/UPDATE(C.Rating)

Search
Page(LandingPage)
Scene (Quick_Search)
View (in) Input-Form(reduced (filterContent))
View (out) Result-List($x | x \in C, filter(x) = true$)

Page(Wiki, Coffee_Shop)
Scene (Direct_Search)
View (in) Input-Form(C.Name++C.Address || A.Name)
View (out) Result(C) || Result(A)

Page(Coffee_Shop,Wiki)
Scene (Elaborate_Search)
View (in) Input-Form(filter)

View (out) Result-List$(x|x \in C||A, filter(x) = true)$

Page(Coffee_Shop,Wiki)
Scene (Sort_by_Result)
View (in) Result-List(C || A)
View (out) Result-List(sort_by(C || A))

Account_Management
Scene (Login)
View (in) Input-Form(U)

Scene (Account)
View (in) Input-Form(U)
View (out) READ/UPDATE(U)

Scene (Registration)
View (in) Input-Form(U)
View (out) INSERT(U)

5 HERM-Schema

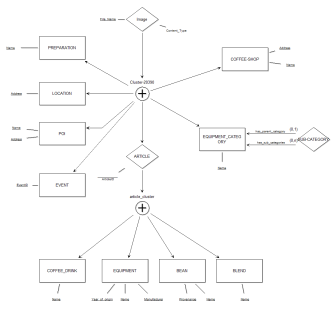

Figure 6: Simplify domain model

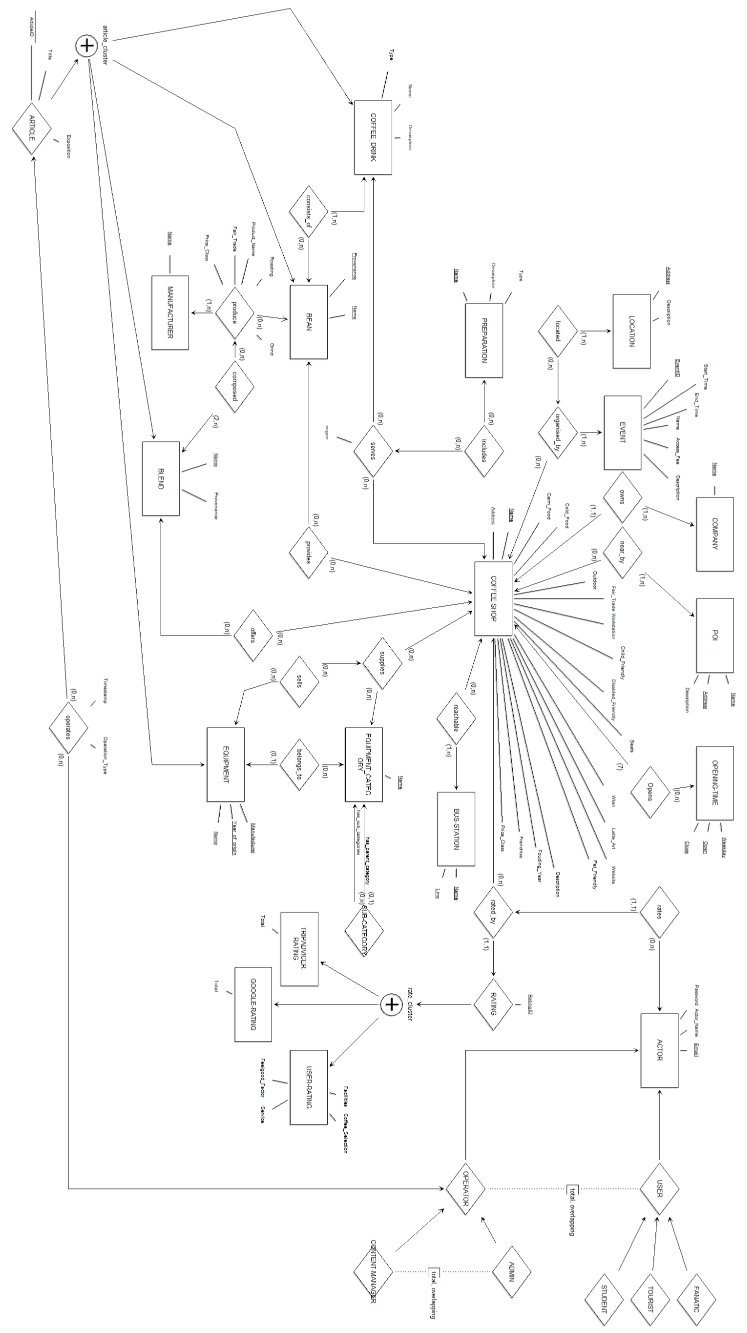

Figure 7: Simplify domain model

5.1 HERM-Translation

We used the *ADOxx* translater to translate the HERM model into the data base schema. Most of it would be decoded without any help, but e.g. the Cluster, we had to translate on our own.

5.1.1 Descripition

Identification

The Identification of the entities and relationship is a combination of using natural keys as well as surrogate keys. The entity COFFEE-SHOP is the most connected entity in our schema but his natural primary key consist of multiple columns. This cause too much cumbersome workarounds to keep the key natural. For that reason the COFFEE-SHOP entity will have in the implementation another attribute id which will be the new primary key. This new key destroy the 3NF of our schema.

Specialisation

All specialisation are total and overlapped.

Higher-Order

Located was translate by taken the primary key of LOCATION as well as the primary keys from the relationship of organised_by.
Rated_By was translated by taken the primary key of RATES, COFFEESHOP and the surrogate key from RATING.
Includes was translate by taken the primary key of PREPARATION as well as the primary keys from the relationship of serves.
Sells was translated by taken the primary key of EQUIPMENT as well as the primary keys from the relationship supplies.

Cluster

The ARTICLE cluster with the connection to the following entites: EQUIPMENT, COFFEE_DRINK, BEAN and BLEND was transformed by using the separation approach where the middle table are collapsed. Seperation was used so that the keys of the connected entites are still the natural one.
The RATING cluster with the connection to the following entities: GOOGLE-RATING, USER-RATING, TRIPADVISER-RATING was transformed by using the full participation approach because there is no common key in the entites.

5.1.2 Entities

(EQUIPMENT(Manufacturer, Year_of_origin, Name)(Manufacturer, Year_-of_origin, Name)),
(EVENT(EventID,Start_Time,Name,Access_Fee,Description,End_Time)(EventID)),
(COFFEE-SHOP(Name, Address, Outdoor, Fair_Trade, Disabled_Friendly, Description, Wlan, Child_Friendly, Website, Fouding_Year, Pet_Friendly, Latte_-Art, Seats, Workstation, Food, Price_Class, Franchise)(Name, Address)),
(BUS-STATION(Name, Line)(Name,Line)),
(COMPANY(Name)(Name)),
(BEAN(Name,Provenance,Type)(Name,Provenance)),
(POI(Name,Address,Description)(Name, Address)),
(GOOGLE-RATING()()),
(USER-RATING()()),
(TRIPADVICER-RATING()()),
(BLEND(Name, Provenance, Price_Range)(Name)),
(LOCATION(Address, Description)(Address)),
(EQUIPMENT_CATEGORY(Name)(Name)),
(ACTOR(Email,Actor_Name, Password)(Email)),
(PREPARATION(Name, Description, Type)(Name)),
(COFFEE_DRINK(Name, Typ, Description)(Name)),
(OPENING-TIME(Close, Open, Weekday)(Close, Open, Weekday)),
(MANUFACTURER(Name)(Name)),
(RATING(RatingID, RATINGId)(RatingID, RATINGId)),

5.1.3 Relationships

(consists_of(Name, Provenance, Name)(Name, Provenance, Name)),
(serves(Name, Address, Name, vegan)(Name, Address, Name)),
(near_by(Name, Address, Name, Address)(Name, Address, Name, Address)),
(reachable(Name, Line, Name, Address)(Name, Line, Name, Address)),
(owns(Name, Address, Name)(Name, Address)),
(supplies(Name, Name, Address)(Name, Name, Address)),
(provides(Name, Address, Name, Provenance)(Name, Address, Name, Provenance)),
(offers(Name, Name, Address)(Name, Name, Address)),
(organised_by(Name, Address, EventID)(Name, Address, EventID)),
(OPERATOR(Email)(Email)),
(SUB-CATEGORY(Name)(Name)),
(belongs_to(Manufacturer, Year_of_origin, Name, Name)(Manufacturer, Year_-of_origin, Name)),
(Opens(Name, Address, Close, Open, Weekday)(Name, Address, Close, Open, Weekday)),
(produce(Name, Provenance, Name, Product_Name, Fair_Trade, Price_Class)(Name, Provenance, Name)),

(includes(Name, Address, Name, Name)(Name, Address, Name, Name)),
(composed(Name, Name, Provenance, Name)(Name, Name, Provenance, Name)),
(rated_by(RatingID, RATINGId, Name, Address)(RatingID, RATINGId)),
(located(Address, Name, Address, EventID)(Address, Name, Address, EventID)),
(sells(Manufacturer, Year_of_origin, Name, Name, Name, Address)(Manufacturer, Year_of_origin, Name, Name, Name, Address)),
(creates(Email, ArticleID)(Email, ArticleID)),
(publishes(Email, ArticleID)(Email, ArticleID)),
(rates(RatingID, RATINGId, Email)(RatingID, RATINGId))

5.1.4 Cluster

(RATINGGOOGLE-RATING(RatingID, RATINGId)(RatingID, RATINGId)),
(RATINGUSER-RATING(RatingID, RATINGId)(RatingID, RATINGId)),
(RATINGTRIPADVICER-RATING(RatingID, RATINGId)(RatingID, RATINGId)),
(ARTICLEEQUIPMENT(ArticleID, Manufacturer, Year_of_origin, Name, Exposition, Title)(ArticleID)),
(ARTICLEBLEND(ArticleID, Name, Exposition, Title)(ArticleID)),
(ARTICLEBEAN(ArticleID, Name, Provenance, Exposition, Title)(ArticleID)),
(ARTICLECOFFEE_DRINK(ArticleID, Name, Exposition, Title)(ArticleID)),

5.1.5 Specialisation

(STUDENT(Email)(Email)),
(TOURIST(Email)(Email)),
(FANATIC(Email)(Email)),
(ADMIN(Email)(Email)),
(CONTENT-MANAGER(Email)(Email)),
(USER(Email)(Email)),

5.1.6 Integrity Constraints

EVENT[EventID]
⊆organised_by[EventID]
BUS-STATION[Name, Line]
⊆reachable[Name, Line]
COMPANY[Name]⊆owns[Name]
BEAN[Name, Provenance]⊆produce[Name, Provenance]
POI[Name, Address]⊆near_by[Name, Address]
LOCATION[Address]⊆located[Address]
COFFEE_DRINK[Name]⊆consists_of[Name]
Manufacutrer[Name]⊆produce[Name]
USER[Email]⊆ACTOR[Email]

consists_of[Name, Provenance]⊆BEAN[Name, Provenance]
consists_of[Name]⊆COFFEE_DRINK[Name]
serves[Name, Address]⊆COFFEE-SHOP[Name, Address]
serves[Name]⊆COFFEE_DRINK[Name]
near_by[Name, Address]⊆COFFEE-SHOP[Name, Address]
near_by[Name, Address]⊆POI[Name, Address]
reachable[Name, Line]⊆BUS-STATION[Name, Line]
reachable[Name, Address]⊆COFFEE-SHOP[Name, Address]
owns[Name, Address]⊆COFFEE-SHOP[Name, Address]
owns[Name]⊆COMPANY[Name]
supplies[Name]⊆EQUIPMENT_CATEGORY[Name]
supplies[Name, Address]⊆COFFEE-SHOP[Name, Address]
provides[Name, Address]⊆COFFEE-SHOP[Name, Address]
provides[Name, Provenance]⊆BEAN[Name, Provenance]
offers[Name]⊆BLEND[Name]
offers[Name, Address]⊆COFFEE-SHOP[Name, Address]
organised_by[Name, Address]⊆COFFEE-SHOP[Name, Address]
organised_by[EventID]⊆EVENT[EventID]
OPERATOR[Email]⊆ACTOR[Email]
SUB-CATEGORY[Name]⊆EQUIPMENT_CATEGORY[Name]
SUB-CATEGORY[Name]⊆EQUIPMENT_CATEGORY[Name]
SUB-CATEGORY[]⊆EQUIPMENT_CATEGORY[]
belongs_to[Name]⊆EQUIPMENT_CATEGORY[Name]
belongs_to[Manufacturer, Year_of_origin, Name]⊆EQUIPMENT[Manufacturer, Year_of_origin, Name]
Opens[Name, Address]⊆COFFEE-SHOP[Name, Address]
Opens[Close, Open, Weekday]⊆Opening-Time[Close, Open, Weekday]
produce[Name, Provenance]⊆BEAN[Name, Provenance]
produce[Name]⊆Manufacutrer[Name]
includes[Name, Address, Name]⊆serves[Name, Address, Name]
includes[Name]⊆PREPARATION[Name]
composed[Name]⊆BLEND[Name]
composed[Name, Provenance, Name]⊆produce[Name, Provenance, Name]
rated_by[Name, Address]⊆COFFEE-SHOP[Name, Address]
rated_by[RatingID, RATINGId]⊆RATING[RatingID, RATINGId]
located[Address]⊆LOCATION[Address]
located[Name, Address, EventID]⊆organised_by[Name, Address, EventID]
sells[Manufacturer, Year_of_origin, Name]⊆EQUIPMENT[Manufacturer, Year_-of_origin, Name]
sells[Name, Name, Address]⊆supplies[Name, Name, Address]
STUDENT[Email]⊆USER[Email]
TOURIST[Email]⊆USER[Email]
FANATIC[Email]⊆USER[Email]
ADMIN[Email]⊆OPERATOR[Email]
CONTENT-MANAGER[Email]⊆OPERATOR[Email]
creates[Email]⊆OPERATOR[Email]

creates[ArticleID]⊆ARTICLEEQUIPMENT[ArticleID]
creates[ArticleID]⊆ARTICLEBLEND[ArticleID]
creates[ArticleID]⊆ARTICLEBEAN[ArticleID]
creates[ArticleID]⊆ARTICLECOFFEE_DRINK[ArticleID]
publishes[Email]⊆OPERATOR[Email]
publishes[ArticleID]⊆ARTICLEEQUIPMENT[ArticleID]
publishes[ArticleID]⊆ARTICLEBLEND[ArticleID]
publishes[ArticleID]⊆ARTICLEBEAN[ArticleID]
publishes[ArticleID]⊆ARTICLECOFFEE_DRINK[ArticleID]
rates[RatingID, RATINGId]⊆rated_by[RatingID, RATINGId]
rates[Email]⊆ACTOR[Email]
ARTICLEEQUIPMENT[ArticleID]
||ARTICLEBLEND[ArticleID]
||ARTICLEBEAN[ArticleID]
||ARTICLECOFFEE_DRINK[ArticleID]

5.1.7 Data Type

EQUIPMENT.Manufacturer::VARCHAR(n) EQUIPMENT.Year_of_origin::VARCHAR(n)
EQUIPMENT.Name::VARCHAR(n)
EVENT.EventID::INTEGER
EVENT.Start_Time::INTEGER
EVENT.Name::VARCHAR(n)
EVENT.Access_Fee::INTEGER
EVENT.Description::VARCHAR(n)
EVENT.End_Time::TIME COFFEE-SHOP.Name::VARCHAR(n)
COFFEE-SHOP.Address::VARCHAR(n)
COFFEE-SHOP.Outdoor::BOOLEAN
COFFEE-SHOP.Fair_Trade::BOOLEAN
COFFEE-SHOP.Disabled_Friendly::BOOLEAN
COFFEE-SHOP.Description::VARCHAR(n)
COFFEE-SHOP.Wlan::BOOLEAN
COFFEE-SHOP.Child_Friendly::BOOLEAN
COFFEE-SHOP.Website::VARCHAR(n)
COFFEE-SHOP.Fouding_Year::INTEGER
COFFEE-SHOP.Pet_Friendly::BOOLEAN
COFFEE-SHOP.Latte_Art::VARCHAR(n)
COFFEE-SHOP.Seats::VARCHAR(n)
COFFEE-SHOP.Workstation::BOOLEAN
COFFEE-SHOP.Food::VARCHAR(n)
COFFEE-SHOP.Price_Class::VARCHAR(n)
COFFEE-SHOP.Franchise::BOOLEAN
BUS-STATION.Name::VARCHAR(n)
BUS-STATION.Line::VARCHAR(n)
COMPANY.Name::VARCHAR(n)

BEAN.Name::VARCHAR(n)
BEAN.Provenance::VARCHAR(n)
BEAN.Type::VARCHAR(n)
POI.Name::VARCHAR(n)
POI.Address::VARCHAR(n)
POI.Description::CHARACTER(n)
BLEND.Name::VARCHAR(n)
BLEND.Provenance::VARCHAR(n)
BLEND.Price_Range::INTEGER
LOCATION.Address::VARCHAR(n)
LOCATION.Description::VARCHAR(n)
EQUIPMENT_CATEGORY.Name::VARCHAR(n)
ACTOR.Email::VARCHAR(n)
ACTOR.Actor_Name::VARCHAR(n)
ACTOR.Password::VARCHAR(n)
PREPARATION.Description::VARCHAR(n)
PREPARATION.Type::VARCHAR(n)
PREPARATION.Name::VARCHAR(n)
COFFEE_DRINK.Typ::VARCHAR(n)
COFFEE_DRINK.Name::VARCHAR(n)
COFFEE_DRINK.Description::VARCHAR(n)
Opening-Time.Close::INTEGER
Opening-Time.Open::INTEGER
Opening-Time.Weekday::VARCHAR(n)
Manufacutrer.Name::VARCHAR(n)
USER.Email::VARCHAR(n)
RATING.RatingID::INTEGER
RATING.RATINGId::INTEGER
consists_of.Name::VARCHAR(n)
consists_of.Provenance::VARCHAR(n)
consists_of.Name::VARCHAR(n)
serves.vegan::BOOLEAN
serves.Name::VARCHAR(n)
serves.Address::VARCHAR(n)
serves.Name::VARCHAR(n)
near_by.Name::VARCHAR(n)
near_by.Address::VARCHAR(n)
near_by.Name::VARCHAR(n)
near_by.Address::VARCHAR(n)
reachable.Name::VARCHAR(n)
reachable.Line::VARCHAR(n)
reachable.Name::VARCHAR(n)
reachable.Address::VARCHAR(n)
owns.Name::VARCHAR(n)
owns.Address::VARCHAR(n)
owns.Name::VARCHAR(n)

supplies.Name::VARCHAR(n)
supplies.Name::VARCHAR(n)
supplies.Address::VARCHAR(n)
provides.Name::VARCHAR(n)
provides.Address::VARCHAR(n)
provides.Name::VARCHAR(n)
provides.Provenance::VARCHAR(n)
offers.Name::VARCHAR(n)
offers.Name::VARCHAR(n)
offers.Address::VARCHAR(n)
organised_by.Name::VARCHAR(n)
organised_by.Address::VARCHAR(n)
organised_by.EventID::INTEGER
OPERATOR.Email::VARCHAR(n)
SUB-CATEGORY.Name::CHAR belongs_to.Manufacturer::VARCHAR(n)
belongs_to.Year_of_origin::VARCHAR(n)
belongs_to.Name::VARCHAR(n)
belongs_to.Name::VARCHAR(n)
Opens.Name::VARCHAR(n)
Opens.Address::VARCHAR(n)
Opens.Close::INTEGER
Opens.Open::INTEGER
Opens.Weekday::VARCHAR(n)
produce.Product_Name::VARCHAR(n)
produce.Fair_Trade::BOOLEAN
produce.Price_Class::CHARACTER(n)
produce.Name::VARCHAR(n)
produce.Provenance::VARCHAR(n)
produce.Name::VARCHAR(n)
RATINGGOOGLE-RATING.RatingID::INTEGER
RATINGGOOGLE-RATING.RATINGId::INTEGER
RATINGUSER-RATING.RatingID::INTEGER
RATINGUSER-RATING.RATINGId::INTEGER
RATINGTRIPADVICER-RATING.RatingID::INTEGER
RATINGTRIPADVICER-RATING.RATINGId::INTEGER
ARTICLEEQUIPMENT.ArticleID::INTEGER
ARTICLEEQUIPMENT.Manufacturer::VARCHAR(n)
ARTICLEEQUIPMENT.Year_of_origin::VARCHAR(n)
ARTICLEEQUIPMENT.Name::VARCHAR(n)
ARTICLEEQUIPMENT.Exposition::CHARACTER(n)
ARTICLEEQUIPMENT.Title::VARCHAR(n)
ARTICLEBLEND.ArticleID::INTEGER
ARTICLEBLEND.Name::VARCHAR(n)
ARTICLEBLEND.Exposition::CHARACTER(n)
ARTICLEBLEND.Title::VARCHAR(n)
ARTICLEBEAN.ArticleID::INTEGER

ARTICLEBEAN.Name::VARCHAR(n)
ARTICLEBEAN.Provenance::VARCHAR(n)
ARTICLEBEAN.Exposition::CHARACTER(n)
ARTICLEBEAN.Title::VARCHAR(n)
ARTICLECOFFEE_DRINK.ArticleID::INTEGER
ARTICLECOFFEE_DRINK.Name::VARCHAR(n)
ARTICLECOFFEE_DRINK.Exposition::CHARACTER(n)
ARTICLECOFFEE_DRINK.Title::VARCHAR(n)
includes.Name::VARCHAR(n)
includes.Address::VARCHAR(n)
includes.Name::VARCHAR(n)
includes.Name::VARCHAR(n)
composed.Name::VARCHAR(n)
composed.Name::VARCHAR(n)
composed.Provenance::VARCHAR(n)
composed.Name::VARCHAR(n)
rated_by.RatingID::INTEGER
rated_by.RATINGId::INTEGER
rated_by.Name::VARCHAR(n)
rated_by.Address::VARCHAR(n)
located.Address::VARCHAR(n)
located.Name::VARCHAR(n)
located.Address::VARCHAR(n)
located.EventID::INTEGER
sells.Manufacturer::VARCHAR(n)
sells.Year_of_origin::VARCHAR(n)
sells.Name::VARCHAR(n)
sells.Name::VARCHAR(n)
sells.Name::VARCHAR(n)
sells.Address::VARCHAR(n)
STUDENT.Email::VARCHAR(n)
TOURIST.Email::VARCHAR(n)
FANATIC.Email::VARCHAR(n)
ADMIN.Email::VARCHAR(n)
CONTENT-MANAGER.Email::VARCHAR(n)
creates.Email::VARCHAR(n)
creates.ArticleID::INTEGER
publishes.Email::VARCHAR(n)
publishes.ArticleID::INTEGER
rates.RatingID::INTEGER
rates.RATINGId::INTEGER
rates.Email::VARCHAR(n)

5.2 Constraints Handling

Referential constraints are enforced through the database management system by adding constraint to the tables which have the corresponding references. The majority of the referential constraints are foreign keys constraints.
Integrity of concrete input of some tables are enforces through checks.

6 Quantity Analysis

The given numbers besides the entity- and relationshiptyps is a guess of the data volume which our database will store.

ENTITY

- SHOP: 55
- USER: 300
- COMPANY: 37
- COFFEE_DRINK: 20
- BLEND: 29
- BEANS: 29
- EVENT: 33
- POI: 28

RELATIONSHIP A CoffeeShop belongs to excatly **1** Company. Also it has **7** OpeningTimes for each day of the week. We assume an average amount of **3** BusStations and **2** POIs for each CoffeeShop. It severs about **30** beverages and offers **10** Beans, **20** Blends and **4** EquipmentCategories. For future profe we provide a sells Table where Shops can offer specific Equipment models, but we do not consider it yet.
A CoffeeShop can organizes several Events, we calculate with a average about **5** Events per year. We do not delete expired Events yet, this might be implemented in later work.

CLUSTER

- ARTICLE: 0
- ARTICLE_BEAN: 0
- ARTICLE_DRINK: 0
- ARTICLE_BLEND: 0

- GOOGLE-RATING:0
- TRIPADVISOR-RATING:0
- USER-RATING:300?

Function Calls Frequency of usage of functionality differentiated between content-manager and regular user:

Content-Manger

- Create: 300 per year
- Update: 1000 per year
- Delete: 50 per year

User

- Search: 400 per day
- Filter: 300 per day

7 BPMN

7.1 Assumptions

Website
: Endpoints are always possible. Therefor we leave out the modeling of it to not overload the models.

Web-Application
: The Web-Application consist of the back-end as well as the front-end. It is always listening for events, for that reason the Web-Application has a unconditional starting point.

Communication
: The communication between front-end(also client) and back-end uses a restful Api through sending JSON datas.

Roles
: The behavior of the Content-Manager is known, from the general Actor not. Therefore the pool in the BPMN Model *CRUD Content* of the Content-Manager is not a black box. The pool in the BPMN Model *Search CoffeeShop from Landingpage* of the Actor is a black box.

7.2 Content-Manager: CRUD Content

The *Content-Manager*(CM) is on the admin area. The CM can now chooses which kind of content{CoffeeShop, Article, Bean, Blend, BusStation, POI, Events } he/she/it wants to processed by clicking on a corresponding tab. The WA gets the choice and renders the response for the CM.
The Content-Manager fills out the Input-Form of the tab and sends it to the WA. It might be a search request for a Content to edit or delete, as well as newly created content. The WA (here the front-end) checks validate the input data. It gives a feedback and let the CM edit the input data to resend it if the input data is not valid. Otherwise, the WA response positive: this could be the Content to edit, or just that the input data is valid.
If the input is not valid the system gives a feedback and let the CM retry the input and awaits a new filled input form. The CM also goes through the loop back and can fill out the input form further. The CM can keep editing and let the WA validate the new input, this process is presented in the *finish?* loop. In the last cycle the CM sends the *safe* command and the WA executes the corresponding data base operation. It sends back a response about the success status of the operation to the CM.
The Content-Manager can chooses other Content to work on or end the Process. The log out process is not model here.

7.3 Actor: Login

The Web-Application is awaiting an *Input-Form* from the *Actor*. After receiving the input, it will check the data with the *DataBase*. Based on the return result it wills reject or accept the *Actor* input and sent the result back to the *Actor*

7.4 Actor: Search CoffeeShop from Landingpage

The modeled search process starts at the Landingpage of the Website. The Web-Application(WA) waits for a search option. At this point it can be a request for *Direct-Search*(Search for a CoffeeShop name) or a *Quick-Search*(Search with reduced input parameters).
For the Direct-Search the input can be just some characters. The WA searches in the data base for all CoffeeShops which starts with the characters. It returns it as list to the Actor. The Actor can change the input and so let the WA search for the changed input. Alternative the Actor can request one of the CoffeeShops from the list. Than the WA gets the CoffeeShop from the data base and sends it to the Actor.
For the Quick-Search the Actor sends a *Input-Form* with parameters. The WA queries the data base for it and returns a list of CoffeeShops, it also presents the Actor the *Elaborate-Search* with all search parameters.
The Actor can now select a CoffeeShop from the list or add further search parameters. If the Actor change search parameters. The WA evaluate the search parameter and sent a list of matching CoffeeShops to the Actor. If the Actor send a select request, the WA send the corresponding shop site back. From this point the Actor can begin a new search with pressing the back button or the *Kaffee* button in the nav bar.

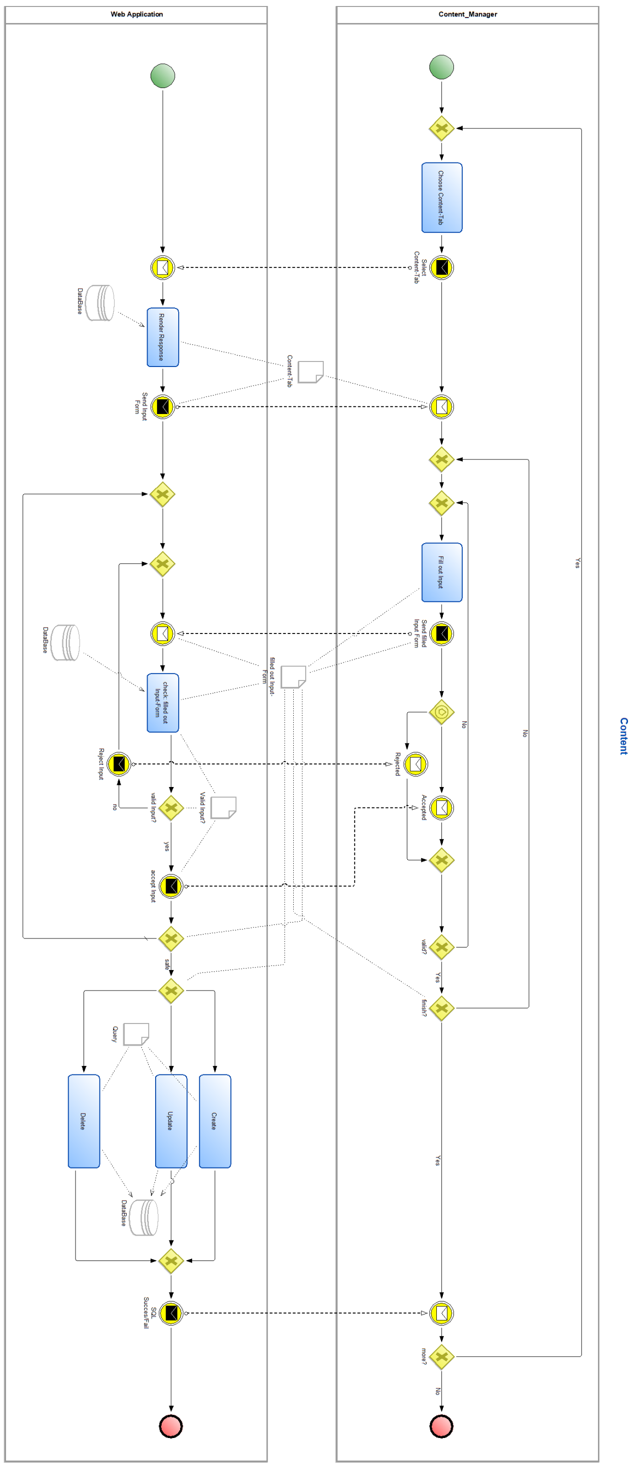

Figure 8: BPMN: Content CRUD Process

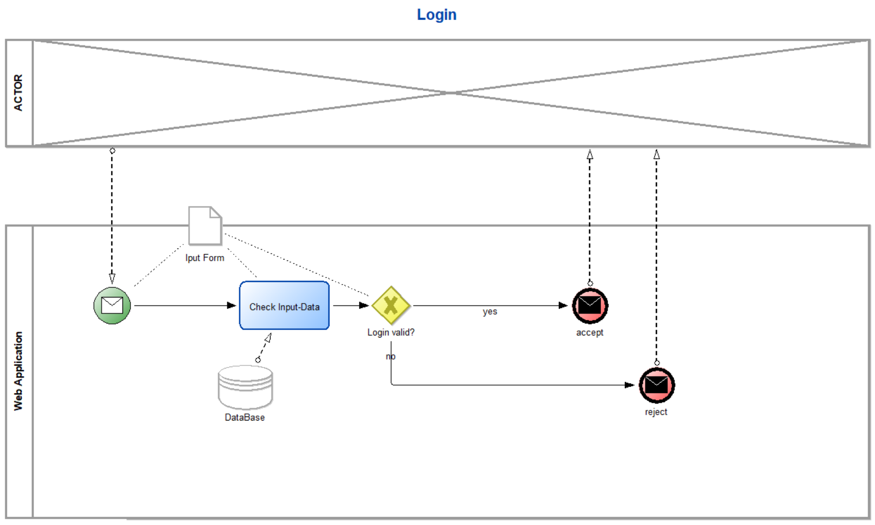

Figure 9: BPMN: Login Process

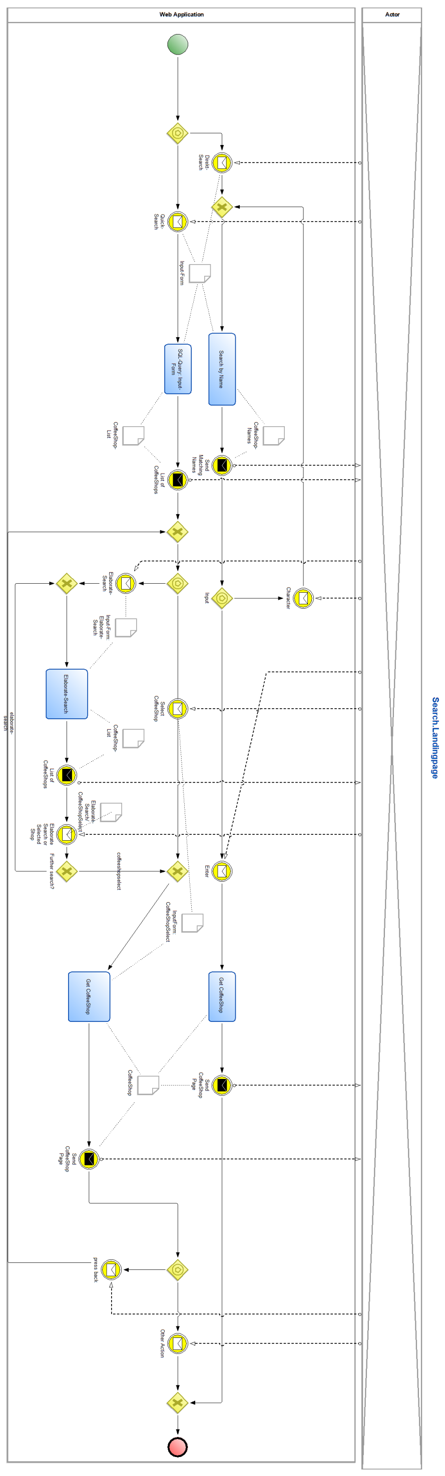

Figure 10: BPMN: Search Process from Landingpage

8 Implementation

- Angular Material
- Angular 7
- ASP.NET Core 2.2
- Dapper
- Identity Framework (with Dapper)
  DB Script and Extension for Identity (to avoid the Entity Framework)
  Source: https://github.com/grandchamp/Identity.Dapper
- POSTGRESQL 11.4
- Docker
- Ubuntu-Server for Deployment (Digital Ocean)

9 Framework

10 Outlook

10.1 External API

To give the user more input of different opinions about a particular shop the web system should include ratings from Google and sites like Trip-Advisor. For a better orientation for the user google map should be include as well as cookies to locate the user current location.

10.2 User Interface / User Experience

General Differents Color Scheming according to seasons? Own original Icons for features of the shops. More Icons to represent different properties. Mobile and Responsive corresponding sites for the general usage. Mobile and Responsive corresponding especially for the input page. **Input Content** Possible to input several contents on one click. Pop up or another method so that it is possible to select as well as input content directly in regards to the shop-tab.

10.3 Behavioral Management

General Can choose how much information is show on. **Fanatics** Make possible to create List of favourites shops. Can Adjust amount of checkboxes in the search box. Make it possible to show selling object of specific shop.