Praktikum zum Modul  
Software Engineering

Sommersemester 2019

Eric Bock: 3265222,

Lars Kähny: 3265612

Lukas Meissner: 3266916,

Annika Netuschil: 3237612

# Kurzbeschreibung des Projekts

Das Ziel unseres Projekts war es, eine Applikation zu bauen, die es einem Nutzer ermöglicht, beim Einkaufen den optimalen Weg durch den Supermarkt zu finden. Somit muss der Nutzer keine Zeit mit unnötigem Suchen von Produkten oder Hin- und herlaufen durch den Supermarkt verschwenden. Außerdem ermöglicht unsere Applikation dem User einen guten Überblick über seine Einkaufsliste zu behalten. Schließlich ist es dem User noch möglich, mit der Applikation einen Überblick über alle Produkte zu erhalten, die in einem Supermarkt verfügbar sind. Das Projekt soll in Form einer Android App mit Xamarin und C# realisiert werden. Der User soll über verschiedene Interfaces Einkaufslisten erstellen können und dann für diese einen Weg durch den von ihm gewählten Supermarkt angezeigt bekommen.

# Anforderungsanalyse

## User Stories

1. Grundnutzen

Der User will seine Zeit im Supermarkt nicht unnötig beim Suchen von Produkten verschwenden, sondern zielgerichtet seinen Einkauf erledigen. In der App erstellt der User eine neue Einkaufsliste. Dort fügt er über den Button "Produkt hinzufügen" neue Produkte hinzu, bis er alle gewünschten Produkte in seiner Einkaufsliste hat. Danach drückt er den Button "Supermarkt auswählen". Jetzt zeigt die App eine Liste mit den verfügbaren Supermärkten an, aus denen der User einen auswählt. Daraufhin erstellt die Software anhand der Einkaufsliste eine Routenplanung für den Einkauf. Diese wird dem User in einer statischen Oberfläche angezeigt. Durch den Button "Einkauf beenden" verlässt man die Oberfläche und kehrt ins Menü zurück.

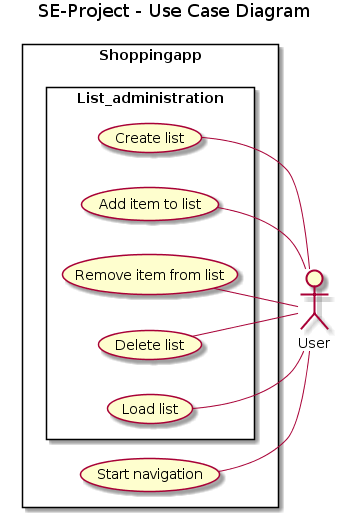
## Anforderungen

1. Das System muss dem Nutzer ermöglichen, den Supermarkt, in dem er einkaufen möchte, einstellen zu können. (Prio 6)
2. Das System muss dem Nutzer ermöglichen, Einkaufsgegenstände mit der gewünschten Anzahl in eine Liste einzufügen, zu entfernen und die Liste zu speichern. (Prio 2)
3. Das System muss anhand der Einkaufsliste den kürzesten Weg durch den Supermarkt für den ausgewählten Supermarkt erstellen können. (Prio 1)
4. Das System muss alle im Supermarkt vorhandenen Produktkategorien kennen. (Prio 7)
5. Das System soll Einkaufsgegenstände bei der Eingabe realen Objekten zuordnen können und erkennen, wenn eine falsche Eingabe getätigt wurde. (Prio 3)
6. Das System soll dem Nutzer nach dem Einkauf ermöglichen, die Liste löschen und speichern zu können. (Prio 4)
7. Das System soll alle Marken der Produktkategorien kennen, um dem Nutzer eine Auswahl der Marke zu ermöglichen. (Prio 8)
8. Das System soll dem Nutzer ermöglichen, gespeicherte Listen als Vorlage für einen neuen Einkauf auszuwählen. (Prio 5)

# Modellierung

## Use Cases (Diagramme und tabellarische Form)

**Use-Case-Diagramm:**



**1. Use Case**

1. Name of use case: Create list

2. Nr.: 1

3. Objective: The User creates a new list with all the items he wants to buy

4. Scope: global

5. Level: main functionality

6. Categorie: primary

7. Involved Classes: Shoppinglist, Product, Supermarket

8. Precondition: User has to have selected the right interface

9. Postcondition (success): List is saved

10. Postcondition (failure): User is returned to main menu. An error message is displayed.

11. Actors: User

12. Trigger: "New List"-Button clicked

13. Actions to exectue:

1. Choose supermarket

2. Enter product categorie

3. Choose product (brand)

4. Add product to list

5. IF not finished: GOTO Step 2

6. Save list

14. Extensions: -

15. Actions in case of exeption:

**2. Use Case (Anforderung 6)**

1. Name of use case: Delete list

2. Nr.: 2

3. Objective: The list the user had selected is deleted.

4. Scope: global

5. Level: secondary functionality

6. Categorie: optional

7. Involved Classes: Shoppinglist

8. Precondition: The user has selected a shoppinglist.

9. Postcondition (success): The selected shoppinglist is deleted

10. Postcondition (failure): The user is informed about an error

11. Actors: user

12. Trigger: "Delet"-Button in list interface is clicked

13. Actions to exectue:

1. The selected list gets deleted

2. The user is returned to the main menu

14. Extensions: A security check can be implemented

15. Actions in case of exeption:

**3. Use Case (Anforderung 2)**

1. Name of use case: Add item to list

2. Nr.: 3

3. Objective: The User can edit add products to the current shoppinglist

4. Scope: global

5. Level: main functionality

6. Categorie: primary

7. Involved Classes: Shoppinglist, Product

8. Precondition: The user has selected a shoppinglist.

9. Postcondition (success): An item is added to the list

10. Postcondition (failure): The list is unchanged

11. Actors: user

12. Trigger: Item "Add"- or "Delet"-Button is pressed

13. Actions to exectue:

1. The User types product category of the product he wants to add

2. A List is shown with all products from that category

3. The user selects one of the Products

4. The Product is added to the current shoppinglist

5. The user is returned to the shoppinglist view

14. Extensions: A suggestion is shown when the user types the product category

15. Actions in case of exeption:

**4. Use Case (Anforderung 8)**

1. Name of use case: Load list

2. Nr.: 4

3. Objective: The User can load one of his former lists

4. Scope: global

5. Level: secondary functionality

6. Categorie: optional

7. Involved Classes: Shoppinglist

8. Precondition: The user has selected the "load list" view in the main menu.

9. Postcondition (success): The selected shoppinglist is loaded and displayed (selected)

10. Postcondition (failure): The user is informed about an error and returned to the main menu

11. Actors: user

12. Trigger: "Load List"-Button clicked

13. Actions to exectue:

1. A list of all saved shoppinglists is displayed

2. The user clicks one of the lists

3. The list is loaded and displayed

14. Extensions: -

15. Actions in case of exeption:

**5. Use Case (Anforderung 3)**

1. Name of use case: Start nagivation

2. Nr.: 5

3. Objective: The user can see the optimal path through the supermarkt he wants to shop in.

4. Scope: global

5. Level: main functionality

6. Categorie: primary

7. Involved Classes: Shoppinglist, NavigatonController, Supermarket

8. Precondition: The user has selected a shoppinglist.

9. Postcondition (success): The Users is shown a map with a highligthed route.

10. Postcondition (failure): The user is informed about an error and returned to the shoppniglist screen

11. Actors: user

12. Trigger: "Start Navigation"-Button in list interface is clicked

13. Actions to exectue:

1. The route is calculated

2. A map is with the route is shown

14. Extensions:

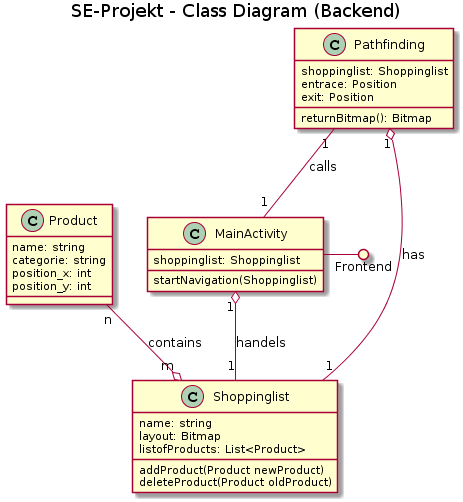
The map show the position of the user.

The map could have a feature to check items that were already bought.

15. Actions in case of exeption:

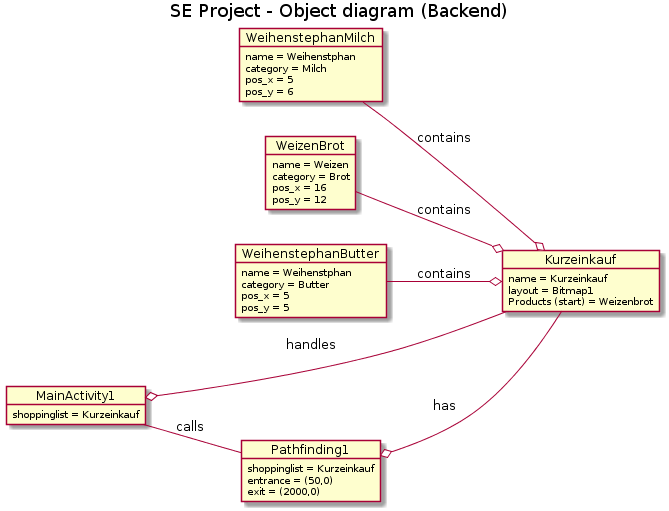
## Klassendiagramme

Klassendiagramm: Backend



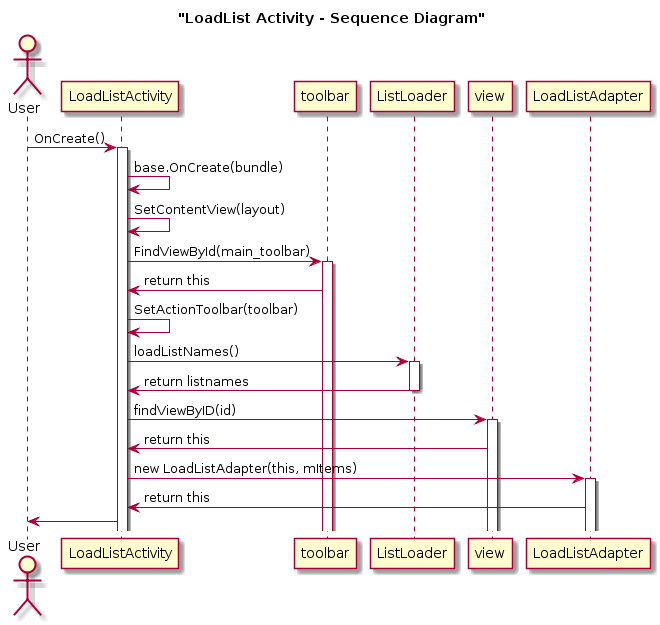
## Objektdiagramme

Objektdiagramm zu Klassendiagramm (Backend)



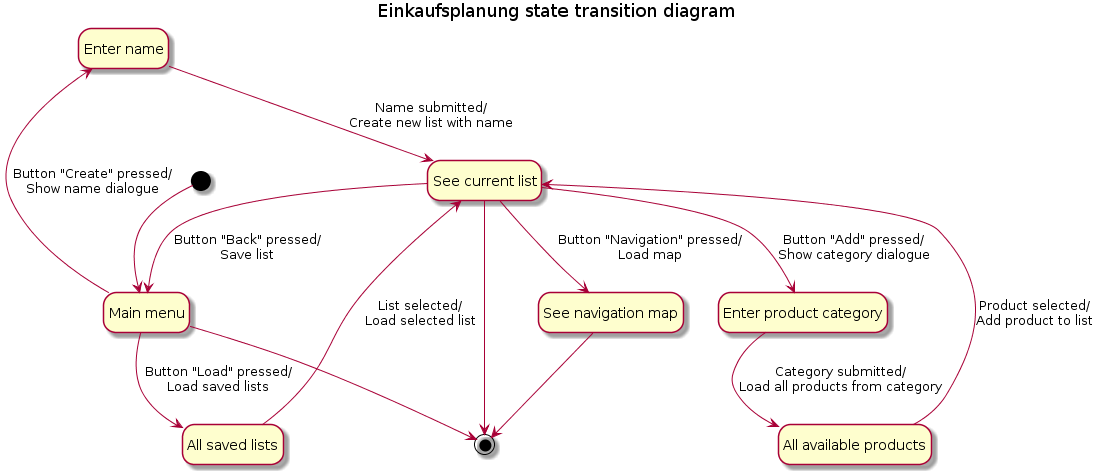
## Sequenzdiagramme

Sequenzdiagramm: LoadlistActivity



## Zustandsdiagramme

Zustandsdiagramm für das User-Interface:



# Implementierung

Zum Entwickeln unserer Applikation haben wir uns für die Programmiersprache C# entschieden. Die Wahl dieser Sprache lag nahe, da wir sie alle im Studium lernen und somit jeder einfach einen Beitrag zur Entwicklung der App leisten konnte.

Als Framework haben wir Xamarin für Visual Studio verwendet. Wir haben Xamarin gewählt, da wir mit der Programmierung in C# bereits vertraut waren und es uns somit naheliegend und einfach erschien.

Zur Versionsverwaltung haben wir GitHub (<https://github.com/Mosman97/SE-Einkaufsplanungs-Application>) genutzt. Es handelt sich hierbei um ein Opensourcetool, zu dem jeder einfach Zugang hat und in dem die einzelnen Versionen und Arbeitsstände einfach gespeichert und geteilt werden können.

Zur Dokumentation und Aufteilung von Aufgaben haben wir „Trello“ (<https://trello.com/b/g7SOBiqE/se-einkaufsplanungs-application>) genutzt. Dort haben wir Aufgaben eingestellt und diese verschiedenen Personen zur Bearbeitung zugeordnet. Außerdem haben wir „Trello“ genutzt, um unsere Arbeit zu dokumentieren und die Abarbeitung der Aufgaben festzuhalten.

Beitrag der einzelnen Teammitglieder:

Lukas Meißner: GUI Programmierung, Verwalten des „Trello“-Boards, Erstellen des Abschlussberichts, Erstellen von Dokumentation und Diagrammen, Erstellen der PowerPoint-Präsentation

Eric Bock:

Lars Kähny: Backend Entwicklung: Implementierung der Wegfindung auf einer Bitmap, in der Klasse „Pathfinding“. Finales Testen der Anwendung sowie fixen von Bugs. Erstellen eines Bitmap-Supermarkts, füllen der Produktdatenbank.

Annika Netuschil: Implementierung der Load List und Store List Methoden, Basiseinführung in XAML, Basisaufbau des Trello-Boards. Teilweises Bugfixing.

## Erfüllte Anforderungen

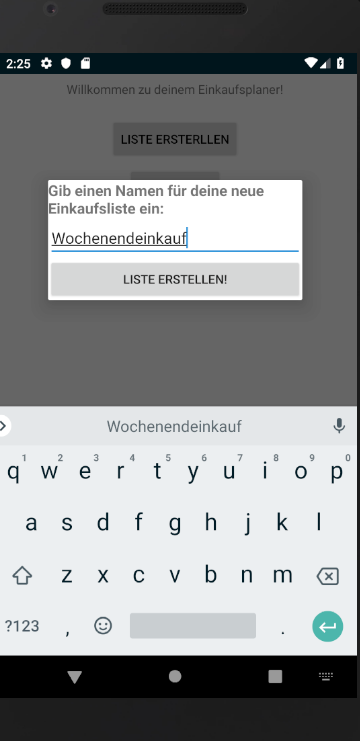
1. Das System muss anhand der Einkaufsliste den kürzesten Weg durch den Supermarkt für den ausgewählten Supermarkt erstellen können. (Anforderung 3, Prio 1) (Es gibt nur einen möglichen Supermarkt)
2. Das System muss dem Nutzer ermöglichen, Einkaufsgegenstände mit der gewünschten Anzahl in eine Liste einzufügen, zu entfernen und die Liste zu speichern. (Anforderung 2, Prio 2) (Es kann keine Anzahl von Produkten angegeben werden)
3. Das System soll Einkaufsgegenstände bei der Eingabe realen Objekten zuordnen können und erkennen, wenn eine falsche Eingabe getätigt wurde. (Anforderung 5, Prio 3) (Falsche Eingaben werden nicht erkannt, aber es werden auch keine Produkte zu falschen Eingaben angezeigt)
4. Das System soll dem Nutzer nach dem Einkauf ermöglichen, die Liste löschen und speichern zu können. (Anforderung 6, Prio 4)
5. Das System soll alle Marken der Produktkategorien kennen, um dem Nutzer eine Auswahl der Marke zu ermöglichen. (Anforderung 7, Prio 8) (Eingetragene Produkte sind Kategorien zugeordnet (Das gesamte Sortiment ist nicht bekannt))
6. Das System soll dem Nutzer ermöglichen, gespeicherte Listen als Vorlage für einen neuen Einkauf auszuwählen. (Anforderung 8, Prio 5)

## Nicht erfüllte Anforderungen

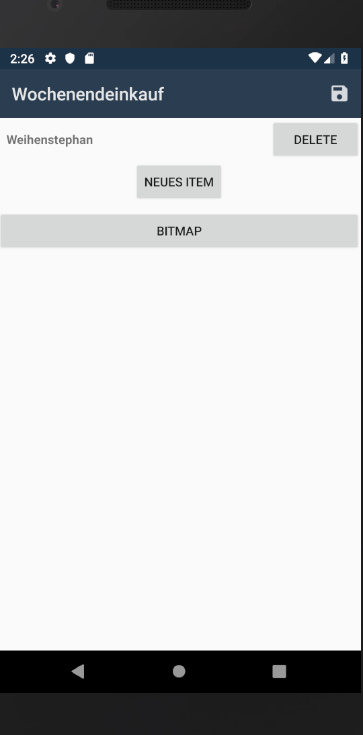
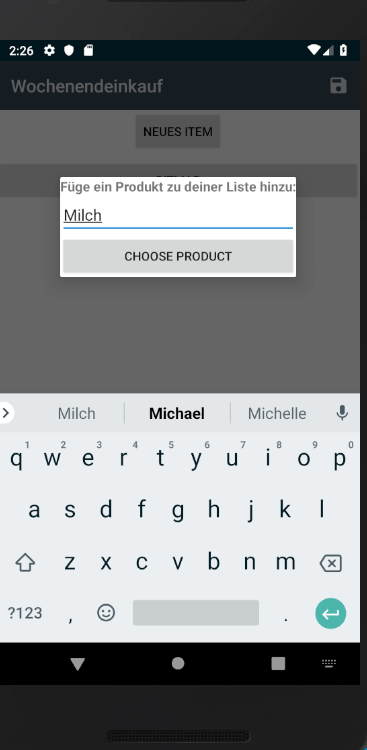
1. Das System muss dem Nutzer ermöglichen, den Supermarkt einstellen zu können, in dem er einkaufen will. (Anforderung 1, Prio 6)
2. Das System muss alle im Supermarkt vorhandenen Produktkategorien kennen. (Anforderung 4, Prio 7)

## User Interface

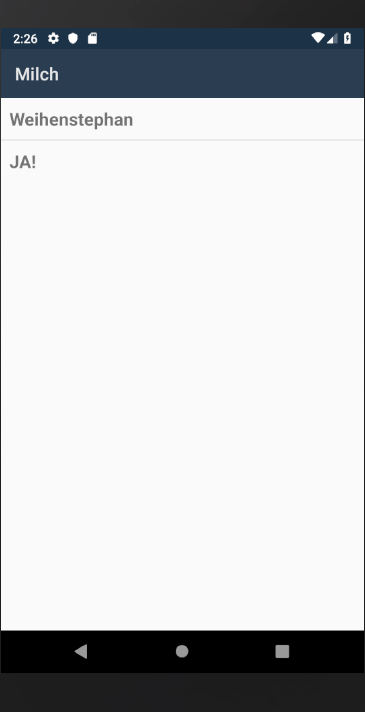
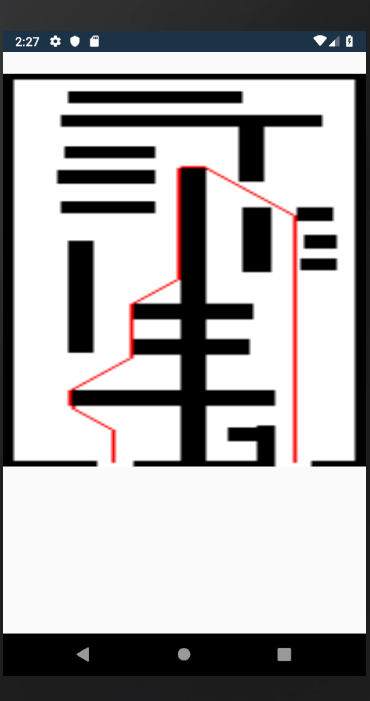
1. Hauptmenü 2. Name eingeben

3. Listenansicht 4. Kategorie auswählen

5. Produkt auswählen 6. Bitmap anzeigen

# Reflexion

Die finale Version unseres Projekts erfüllt unsere Grundidee (Wegfindung durch einen Supermarkt). Wir haben es jedoch nicht geschafft, unsere App praxistauglich zu machen, da wir Funktionen, wie zum Beispiel Supermarktauswahl oder einen Produktkatalog, nicht umsetzen konnten. Es war uns jedoch von Anfang an bewusst, dass eine Umsetzung dieser Funktionen für unser kleines Team mit nur beschränkter Zeit kaum möglich sein würde. Trotzdem erfüllt das Projekt weitestgehend unsere ursprünglichen Erwartungen.

Unsere ursprüngliche Planung stimmt kaum mit der tatsächlich implementierten Lösung überein. In der Planungsphase waren uns viele Restriktionen und Probleme des Implementierens nicht bewusst. Deshalb mussten wir vor allem unser Klassendiagramm öfter überarbeiten bzw. neugestalten.

Die Umsetzung eines realen Supermarkts in die App hat uns an eine Grenze gebracht, die wir nicht überwinden konnten. Der Aufwand, das komplette Sortiment in die App zu übernehmen und den Grundriss in eine Bitmap mit den Produkten an den richtigen Stellen zu übertragen, war zu groß. Deshalb haben wir nur ein kleines ausgewähltes Sortiment mit einem imaginären Supermarkt erstellt und umgesetzt. Hier könnte in einer kommerziellen Version der App vielleicht die Supermärkt mithelfen, indem Sie den Grundriss und das Sortiment ihres Supermarkts pflegen und über eine Schnittstelle zur Verfügung stellen.

Die Arbeitsteilung hat weitestgehend gut funktioniert. Leider ist der Arbeitsfortschritt an manchen Stellen langsamer als erwartet fortgeschritten. Deshalb mussten wir zum Teil auf den Abschluss einzelner Aufgaben warten, bis ein weiterer Teil angefangen werden konnte. Trotzdem hat sich das Projekt durchgehend gut und schnell entwickelt.

Im nächsten Projekt würden wir uns vor allem im Voraus mehr über die Tools informieren, die wir verwenden wollen, um deren Möglichkeiten optimal auszunutzen. Ein Beispiel dafür ist GitHub. Zuerst war uns nicht klar, wie genau wir welche Teile des Projekts einchecken sollen. Deshalb haben wir das gesamte Projekt, anstatt nur den Source-Files, eingecheckt. Dieser Fehler ist uns erst im Nachhinein aufgefallen, als wir Probleme mit dem Pullen/Pushen von Änderungen hatten. Des Weiteren hat uns Xamarin Probleme bereitet, da wir eine weniger weit verbreitete Version verwendet haben, für die es nur wenige Anleitungen und Guides gab, die uns weiterhelfen konnten.