HAWAI-AST

im Auftrag der Firma

Hochschule für Angewandte Wissenschaften Hamburg (HAW Hamburg)
Berliner Tor 5
20099 Hamburg
Deutschland

Architektur

Fabian Pfaff

Version: 0.1
Status: In Arbeit
Stand: 04.05.2015

Zusammenfassung

Dieses Dokument beschreibt die Architektur des HAWAI-AST.

Historie

Version	Status	Datum	Autor(en)	Erläuterung
0.1	In Arbeit	04.05.2015	Fabian Pfaff	Initiale Version für das SEP2 im 4. Semester

Inhaltsverzeichnis

1 Einleitung

1.1 Ziele

Ziel dieses Dokumentes ist die Beschreibung der Architektur des HAWAI-AST, sowie deren Infrastruktur und grundlegenden Konzepte.

1.2 Konventionen

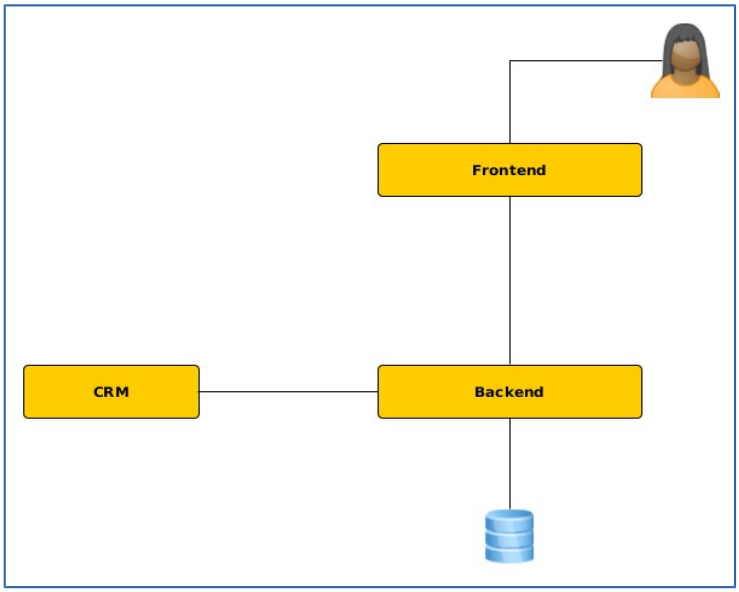
Eigennamen werden in Anführungszeichen geschrieben:
„Klassenname“

Sourcecode wird folgendermaßen dargestellt, Schlüsselwörter und Typen sind dabei farbig markiert:

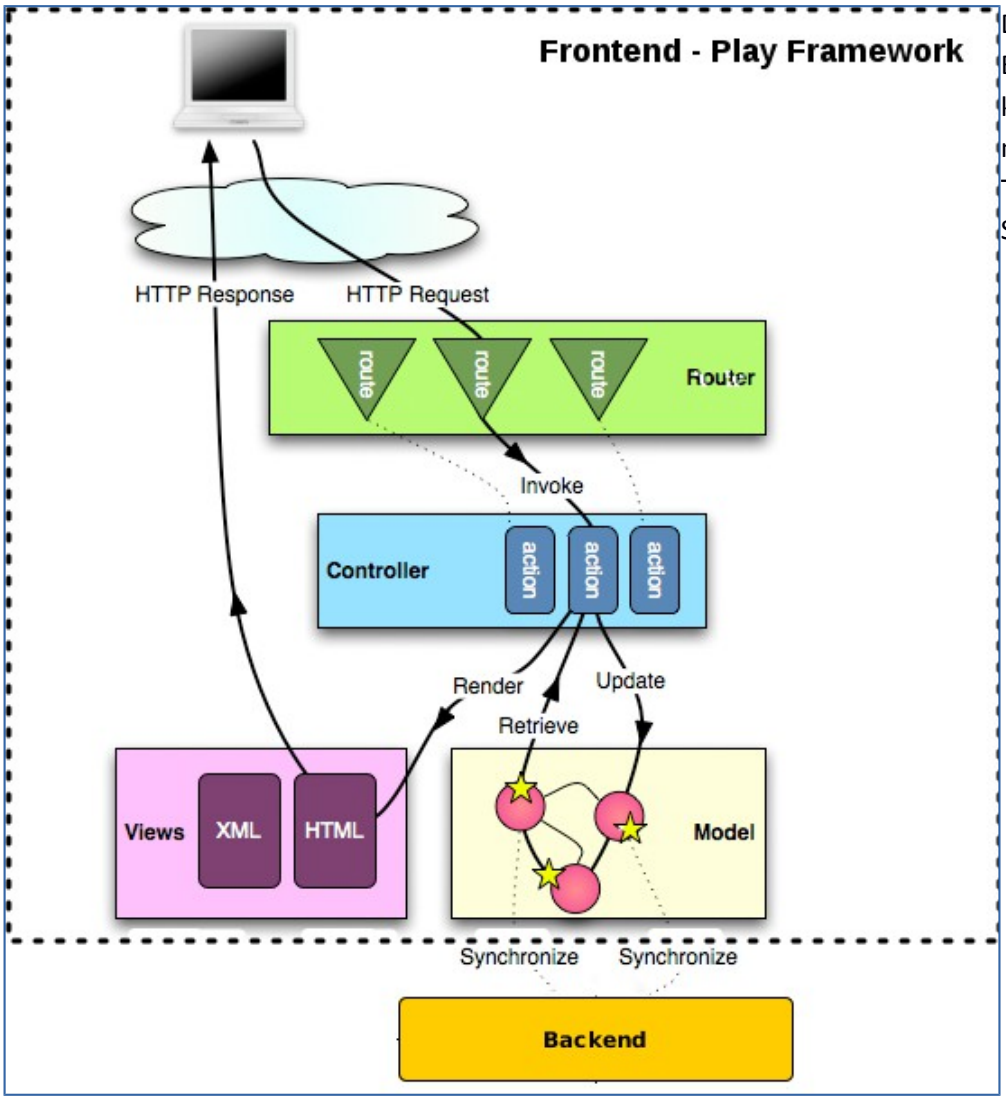
```
int createKunde(String name);
```

2 Architektur des Gesamtsystems

In diesem Kapitel wird die Architektur des HAWAI-AST beschrieben.



Frontend

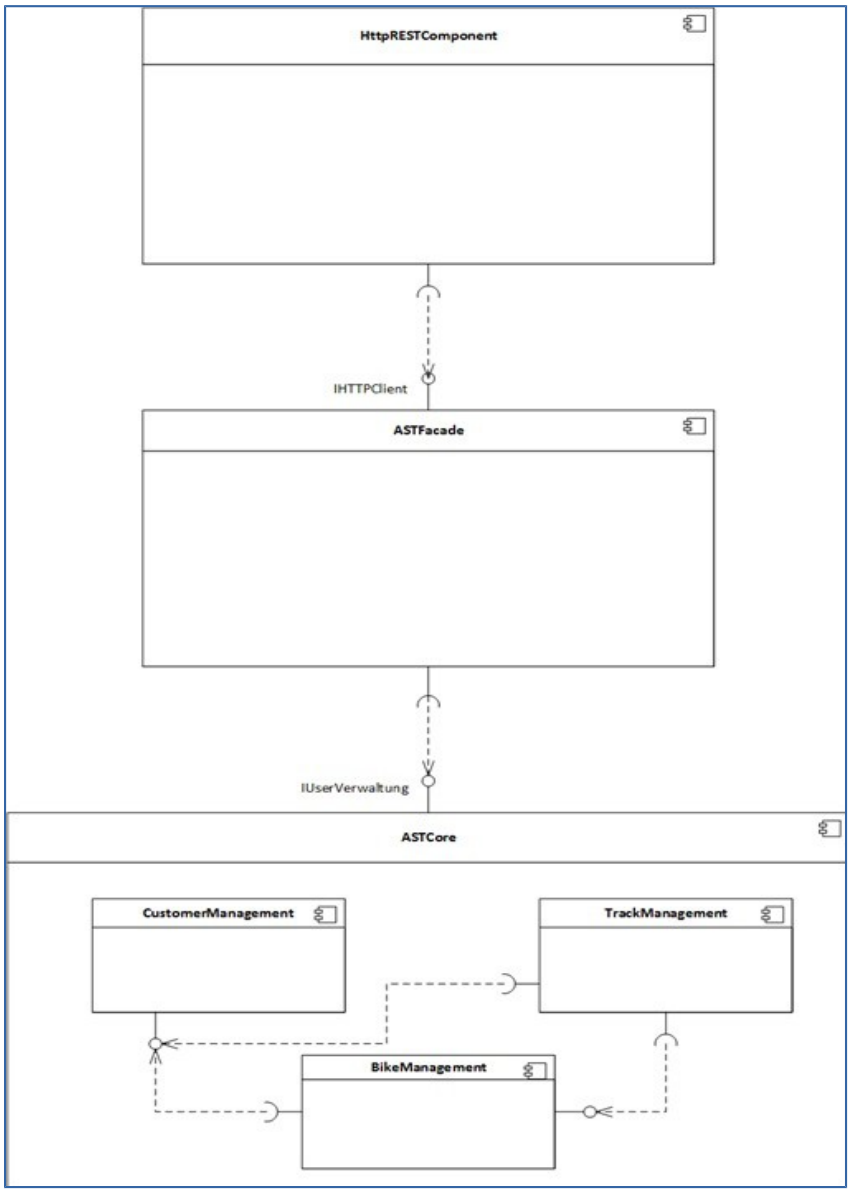


Der Endnutzer kann nur mit diesem Teil der Software kommunizieren.
Das Frontend wird mir Hilfe des Play-Frameworks erstellt und dient dem Bereitstellen einer Weboberfläche als graphische Benutzerschnittstelle.
Das Frontend kommuniziert über eine REST-Schnittstelle mit dem Backend.

Backend

Das Backend dient als Zwischenschicht zur Persistenz, bestehend aus einer SQL-Datenbank und dem CRM „SuiteCRM“. Das Backend wird mit Hilfe des Spring-Frameworks umgesetzt.

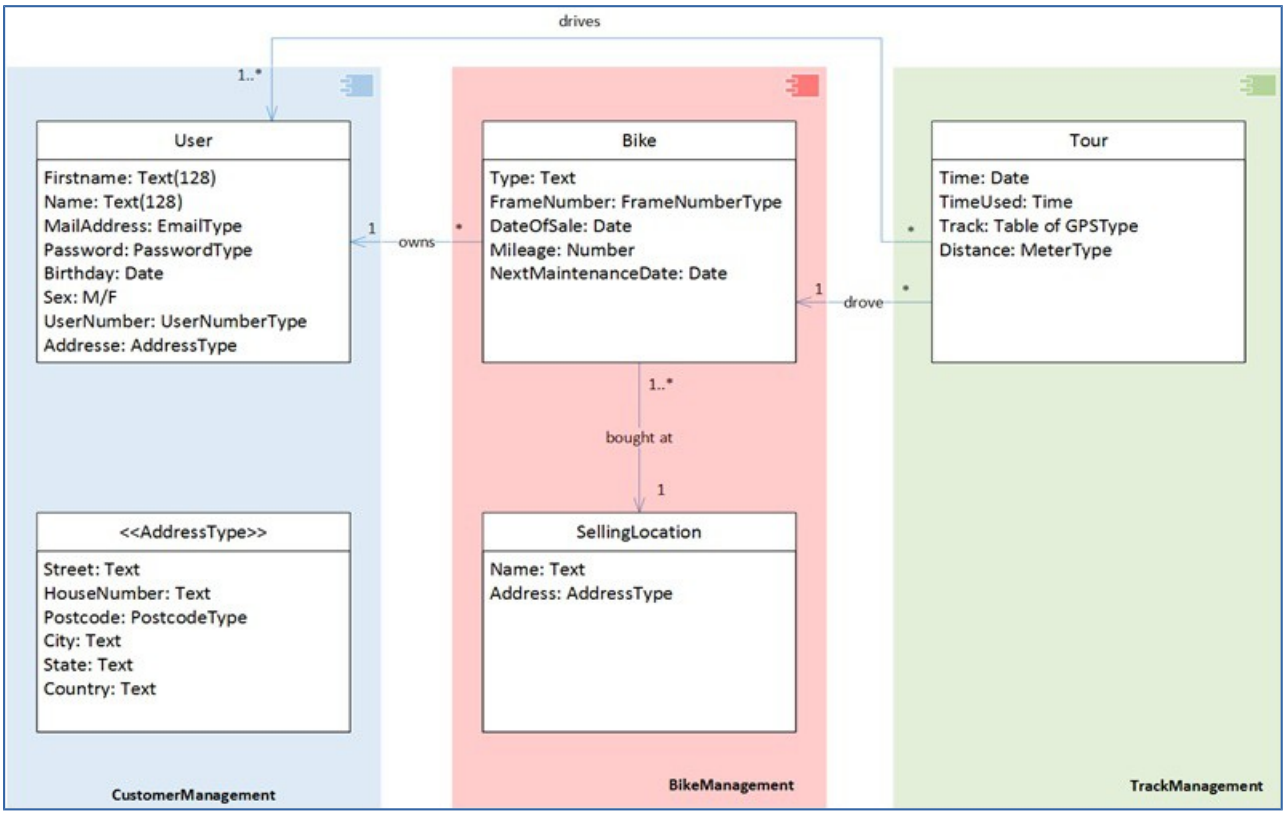
Komponentenschnitt Backend:



Nur die „HTTPRestComponent“ kommuniziert mit der Außenwelt.
Die Facade ist eine Zwischenschicht zwischen rein internen Komponenten („ASTCore“) und den Komponenten, die Schnittstellen nach außen anbieten.

Datenmodel

Im Datenmodel ist die Zugehörigkeit der einzelnen Datentypen zu ihrer Komponente im Backend durch farbliche Hinterlegung gekennzeichnet.


drives
1..*
User
Firstname: Text(128)
Name: Text(128)
MailAddress: EmailType
Password: PasswordType
Birthday: Date
Sex: M/F
UserNumber: UserNumberType
Addresse: AddressType
1
owns
*
Bike
Type: Text
FrameNumber: FrameNumberType
DateOfSale: Date
Mileage: Number
NextMaintenanceDate: Date
1..*
bought at
1
SellingLocation
Name: Text
Address: AddressType
*
1
drove
*
Tour
Time: Date
TimeUsed: Time
Track: Table of GPSType
Distance: MeterType
<<AddressType>>
Street: Text
HouseNumber: Text
Postcode: PostcodeType
City: Text
State: Text
Country: Text
CustomerManagement
BikeManagement
TrackManagement

3 Komponenten

3.1 Komponente CustomerManagement

3.1.1 Verantwortungen der Komponente

Die CustomerManegementComponent verwaltet den Zugriff auf die Benutzerkonten und deren Persistenz.

3.1.2 Außensicht

```
public interface ICustomerManagement {

    public Optional<IUser> getUserBy(EMail eMail);

    public IUser registerUser(String name, String firstName,
            EMail eMailAddress, Address address, Date birthdate, String password);

}
```

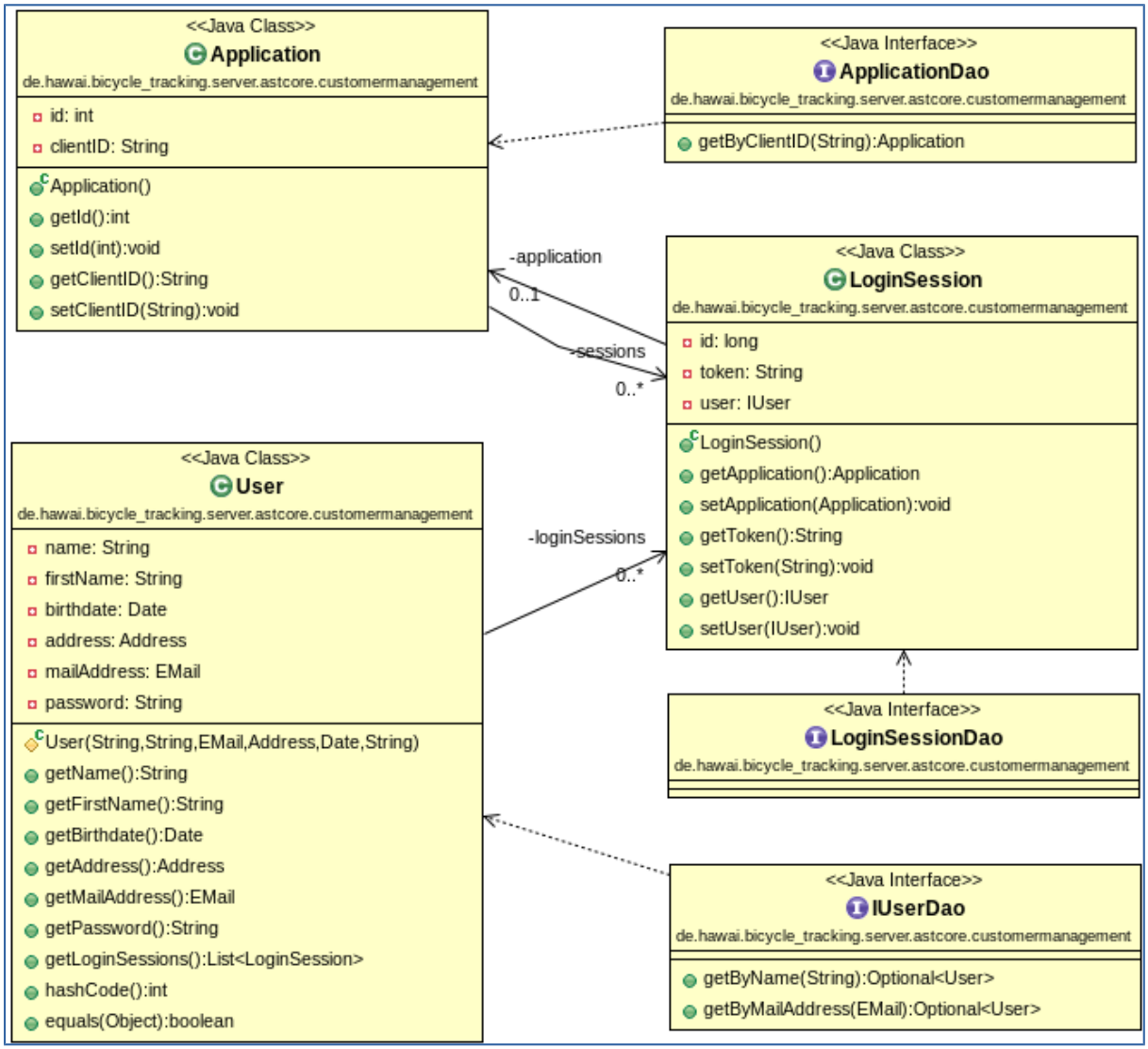
`public Optional<IUser> getUserBy(EMail eMail):`

Gibt den Benutzer mit der gegebenen Email zurück.

`public IUser registerUser(String name, String firstName, EMail eMailAddress, Address address, Date birthdate, String password):`

Erstellt einen neuen Benutzer mit den gegebenen Attributen und persistiert diesen.

3.1.3 Innensicht



Die Entität „User“, die einen Benutzer der Software repräsentiert bekommt bei einem Login eine „LoginSession“ zugeordnet.

Eine „LoginSession“ wird einer „Application“ zugeordnet, um Sessions, die von verschiedenen Clients gestartet wurden zu unterscheiden.

3.1.4 Entwurfsentscheidungen

3.1.5 Schnittstellen zu Nachbarsystemen

Keine.

3.2 Komponente IBikeManagementComponent

3.2.1 Verantwortungen der Komponente

Die „BikeManagement“-Komponente verwaltet die von Nutzern erstellten Fahrräder.

3.2.2 Außensicht

```
public interface IBikeManagement {

    public List<? extends IBike> findBikesBySoldLocation(ISellingLocation inSellingLocation);

    public List<? extends IBike> findByOwner(IUser inOwner);

    public IBike createBike(String inType, FrameNumber inFrameNumber, Date inBuyDate,
            Date inNextMaintenanceDate, ISellingLocation inSellingLocation, IUser inOwner);

    public ISellingLocation createSellingLocation(Address inAddress, String inName);

}
```

`public List<? extends IBike> findBikesBySoldLocation(ISellingLocation inSellingLocation):`

Gibt alle an einem Verkaufsstandpunkt verkauften Fahrrad zurück.

`public List<? extends IBike> findByOwner(IUser inOwner):`

Gibt alle Fahrräder eines Nutzers zurück.

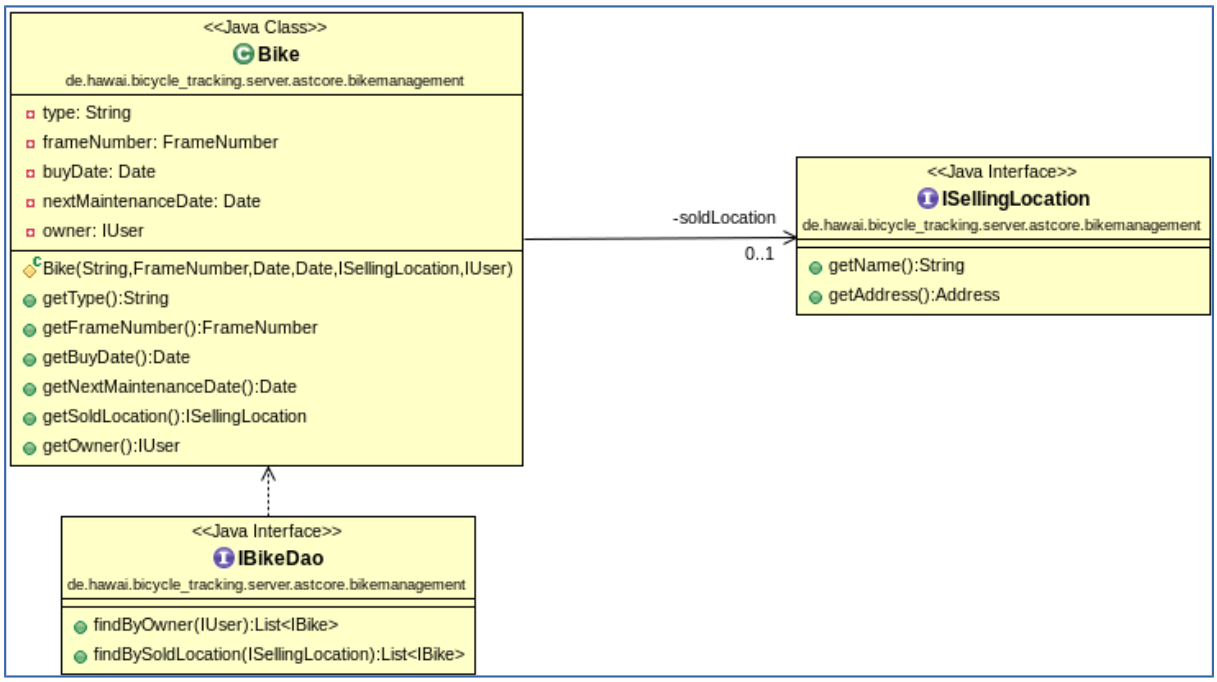
`public IBike createBike(String inType, FrameNumber inFrameNumber, Date inBuyDate, Date inNextMaintenanceDate, ISellingLocation inSellingLocation, IUser inOwner):`

Erstellt ein Fahrrad mit den gegebenen Attributen und persistiert dieses.

`public ISellingLocation createSellingLocation(Address inAddress, String inName):`

Erstellt einen Verkaufsstandort mit den gegebenen Attributen und persistiert diesen.

3.2.3 Innensicht



Die Entität „Bike“ repräsentiert ein Fahrrad, das ein Kunde über die Weboberfläche angelegt hat.

Einem „Bike“ kann eine „SellingLocation“ (Verkaufsort) zugeordnet werden.

3.2.4 Entwurfsentscheidungen

3.2.5 Schnittstellen zu Nachbarsystemen

Keine.

3.3 Komponente HTTPRestComponent

3.3.1 Verantwortungen der Komponente

Die HTTPRest-Komponente stellt die Schnittstellen zu Nachbarsystemen bereit.
Die Spezifikation der Schnittstelle findet sich im Anhang.

4 Infrastruktur

Dieses Kapitel beschreibt die Infrastrukturkomponenten und -vorgaben des Projekts.

4.1 Persistenz

Die Daten werden in einem externen CRM-System persistiert.
Zur schnelleren Bearbeitung von Anfragen soll das Backend jedoch eine lokale Kopie auf einer Relationalen Datenbank vorhalten.

4.2 Logging und Tracing

4.3 Konfiguration

Das Frontend kommuniziert per JSON-Paketen mit dem Backend.
Dabei identifiziert es sich ihm gegenüber durch den Headereintrag („Client-ID“; „DEV-101“).

5 Offene Punkte

- Festlegung auf eine spezifische Datenbank
- Spezifikation der Schnittstelle zum CRM-System
- Caching-Konzept für Daten aus dem CRM im Backend

6 Literatur