1. Änderungen FindLunch für iOS-App

1.1. Deaktivieren SSL

Da Server momentan nicht über das Internet erreichbar ist und die iOS-App nicht ohne weiteres selbstsignierten SSL-Zertifikaten vertraut, wurde die SSL-Verschlüsselung der Web-App vorerst deaktiviert. Bei abermaligen hosten des Servers über eine statische IP-Adresse muss ggf. die iOS-App nochmals geprüft werden, ob die REST-Dienste auch über das HTTPS-Protokoll problemlos laufen.

Deaktivierung SSL in der Datei *application.properties*:

Den Abschnitt „Embedded tomcat configuration“ wie folgt ändern:

```
# Embedded tomcat configuration
server.port=8080
#server.ssl.key-store=classpath:keystore.jks
#server.ssl.key-store-password=f1ndLunch_SSL
#server.ssl.key-password=f1ndLunch_SSL
```

1.2. Hinzufügen zusätzlicher REST-Methode

Es wurde eine zusätzliche REST-Methode für die Rückgabe aller favorisierten Restaurants. Damit diese erreichbar ist musste die URI der Methode (*api/fav_restaurants*) in der Klasse *SecurityConfig.java* hinzugefügt werden. Die Methode sieht nach den Änderungen wie folgt aus:

```
@Override
protected void configure(HttpSecurity http) throws Exception {
  http.csrf().disable().requestMatchers()
       .antMatchers(HttpMethod.GET, "/api/login_user")
       .antMatchers(HttpMethod.POST, "/api/register_push")
       .antMatchers(HttpMethod.GET, "/api/get_push")
       .antMatchers(HttpMethod.DELETE, "/api/unregister_push/**")
       .antMatchers(HttpMethod.PUT, "/api/register_favorite/**")
       .antMatchers(HttpMethod.DELETE, "/api/unregister_favorite/**")
       .antMatchers(HttpMethod.GET, "/api/restaurants")
       .antMatchers(HttpMethod.GET, "/api/fav_restaurants")
     .and().httpBasic().and().sessionManagement()
     .sessionCreationPolicy(SessionCreationPolicy.STATELESS);
}
```

Die Methode muss nun noch in der Klasse *RestaurantRestController.java* hinzugefügt werden:

```
/**
 * Gets all favorized restaurants of a user
 * @param principal
 * @param request
 * @return favorized restaurants of user
 */
@JsonView(RestaurantView.RestaurantRest.class)
@PreAuthorize("isAuthenticated()")
@RequestMapping(path = "/api/fav_restaurants", method = RequestMethod.GET,  headers = {
"Authorization" })
public List<Restaurant> getFavRestaurants(
     @RequestParam(name = "longitude", required = true) float longitude,
     @RequestParam(name = "latitude", required = true) float latitude,
     Principal principal, HttpServletRequest request) {
  LOGGER.info(LogUtils.getInfoStringWithParameterList(request,
```

```
Thread.currentThread().getStackTrace()[1].getMethodName()));

  User authenticatedUser = (User) ((Authentication) principal).getPrincipal();

  // Check favorites
  List<Restaurant> favorites =
restaurantRepo.findByFavUsers_username(authenticatedUser.getUsername());
  for(Restaurant restaurant : favorites){
    restaurant.setDistance(
        DistanceCalculator.calculateDistance(latitude,
            longitude,
            restaurant.getLocationLatitude(),
            restaurant.getLocationLongitude()));
  }

  return favorites;
}
```

1.3. Erweiterung Rückgabedaten

1.3.1. DayOfWeek

Damit die Informationen der Angebotszeiten zusätzlich zurückgegeben werden, mussten die Annotationen der entsprechenden Felder der Domain-Klasse *DayOfWeek.java* angepasst werden.

```
/** The id. */
@Id
@GeneratedValue(strategy=GenerationType.AUTO)
@JsonView({OfferView.OfferRest.class, RestaurantView.RestaurantRest.class,
PushNotificationView.PushNotificationRest.class})
private int id;

/** The day number. */
@Column(name="day_number")
@JsonView({OfferView.OfferRest.class, RestaurantView.RestaurantRest.class,
PushNotificationView.PushNotificationRest.class})
private int dayNumber;

/** The name. */
@JsonView({OfferView.OfferRest.class, RestaurantView.RestaurantRest.class,
PushNotificationView.PushNotificationRest.class})
private String name;
```

1.3.2. Offer

Gleiches gilt für drei Felder der Angebotsklasse *Offer.java*.

```
/** The end date. */
@Temporal(TemporalType.DATE)
@DateTimeFormat(pattern="dd.MM.yyy")
@Column(name="end_date")
@NotNull(message="{offer.endDate.notNull}")
@JsonView(OfferView.OfferRest.class)
private Date endDate;
```

```
/** The start date. */
@Temporal(TemporalType.DATE)
@DateTimeFormat(pattern="dd.MM.yyy")
@Column(name="start_date")
@NotNull(message="{offer.startDate.notNull}")
@JsonView(OfferView.OfferRest.class)
private Date startDate;
```

```
/** The day of weeks. */
//bi-directional many-to-many association to DayOfWeek
@ManyToMany
@JoinTable(
  name="offer_has_day_of_week"
  , joinColumns={
    @JoinColumn(name="offer_id")
    }
  , inverseJoinColumns={
    @JoinColumn(name="day_of_week_id")
    }
  )
@JsonView(OfferView.OfferRest.class)
private List<DayOfWeek> dayOfWeeks;
```

1.3.3. Sender-ID für Firebase GCM

In der iOS-App werden Pushbenachrichtigungen über Google Firebase versendet. Damit dies korrekt funktioniert, muss ein dementsprechendes Projekt auf *firebase.google.com* erzeugt werden. Dessen ID muss letztendlich in der Datei *PushNotificationScheduledTask.java* innerhalb der Unterklasse *SendPushNotification* als *SENDER_ID* hinterlegt werden. Bspw. so:

```
private final String SENDER_ID = "AAAA6u8NA5U:APA91bFJXyb6fYPsBkNNrpy-vXTae-
r5zrNoMvo7uNysQmCAdEU2RLNuHNXghXD8F2B-
yNXJ2ARb_ceNvgtDPiJMnQAjAl3QFjRhowo0FZPmysNcNSrudMRhDn0TTptq3Crw-rb1nSoJ";
```