UNIVERSITATEA POLITEHNICA BUCUREŞTI

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

|  |  |
| --- | --- |
| upb | cs |

Integrarea şi managementul serviciilor

Holiday Advisor – Etapa II

|  |  |
| --- | --- |
|  | Studenti:  Morosan Adrian Gabriel, SSA  Sinziana Nicolae, SSA  Radu Ionut Marian, SSA |

1. **Tehnologii folosite**

Partea de backend este scrisa în Java 8, folosind framework-ul Spring. Pentru expunerea resurselor, se foloseste Spring REST. Resursele sunt expuse de catre server, la o adresa unica. Salvarea informatiilor de la utilizator se va face de catre Watson, in contextul specific fiecarui utilizator, prin interpretarea mesajelor de la acesta.

Arhitectura aplicatiei de backend este impartita pe 3 nivele:

* web: expune resursele pentru public, pe internet
* servicii: logica aplicatiei (parsarea request-urilor, apelarea serviciilor)
* date: salvarea informatiilor de la utilizator (nume, oras, data excursiei, id-ul conversatiei)

Aplicatia web este realizata in Javascript cu framework-ul AngularJS si HTML (HyperText Markup Language), CSS (Cascading Style Sheets) alaturi de framework-ul Bootstrap. Alegerea a fost realizata luand in considerare avantajele pe care le expun cele doua framework-uri precum:

* AngularJS este cel mai actual si popular framework ce se bazeaza pe arhitectura model-view-controller (MVC) ce poate fi folosit in paralel cu jQuery extinzandu-i capabilitatile. Permite prelucrarea datelor, dinamizarea, afişarea si filtrarea direct pe partea client
* Bootstrap este un framework pentru design HTML/CSS și Javascript prin încapsularea unor funcționalități din jQuery ce faciliteaza si imbunatateste procesul de realizare a componentelor unei pagini HTML.

Logica aplicatiei web se bazeaza pe interactiunea dintre utilizator prin introducerea de raspunsuri la intrebarile primite de la Watson si pe afisarea raspunsurilor aferente prin apelarea API-ului pus la dispozitie de catre partea de backend.

Aplicatia de mobile este dezvoltata exclusiv pentru Android si a fost realizata folosind Android SDK 23, putand fi rulat incepand de la versiunea 15 (Ice cream sandwich).

Pentru a se putea afisa harta de la Google Maps, s-a folosit un serviciu care necesita inregistrarea aplicatiei in Google Developers Console si folosirea unui api\_key generat pe baza ei.

Dezvoltarea aplicatiei a fost realizata in Android Studio, iar testarea a fost facuta atat pe device-uri fizice (eg. LG G3, LG Pad 7), cat si pe emulatoare - Android Emulator (Nexus 5).

1. **Serviciile integrate**

Holiday Advisor este o aplicatie care ofera un serviciu interactiv prin care utilizatorii pot gasi repede si usor informatii despre locurile sale preferate sau despre locuri noi pe care vrea sa le viziteze. Proiectul integreaza 4 servicii externe:

**1. IBM Watson Conversation**

Se ocupa de convorbirea dintre utilizator si bot:

- cu ajutorul lui, se extrag informatii importante din conversatia cu utilizatorul (numele lui, numele orasului pe care vrea sa-l viziteze, perioada in care vrea sa calatoreasca etc.)

- acesta este configurat sa urmeze anumiti pasi, pana la oferirea tuturor informatiilor cerute de utilizator

**2. Google Places**

Folosit pentru cautarea anumitor locatii, in functie de orasul si dorinta utilizatorului, precum si subcategoria selectata de acesta. Acesta intoarce o lista de 15 locatii pentru acea subcategorie, impreuna cu detalii despre locatie: nume, adresa, pozitionare pe glob etc.

**3. Google Maps**

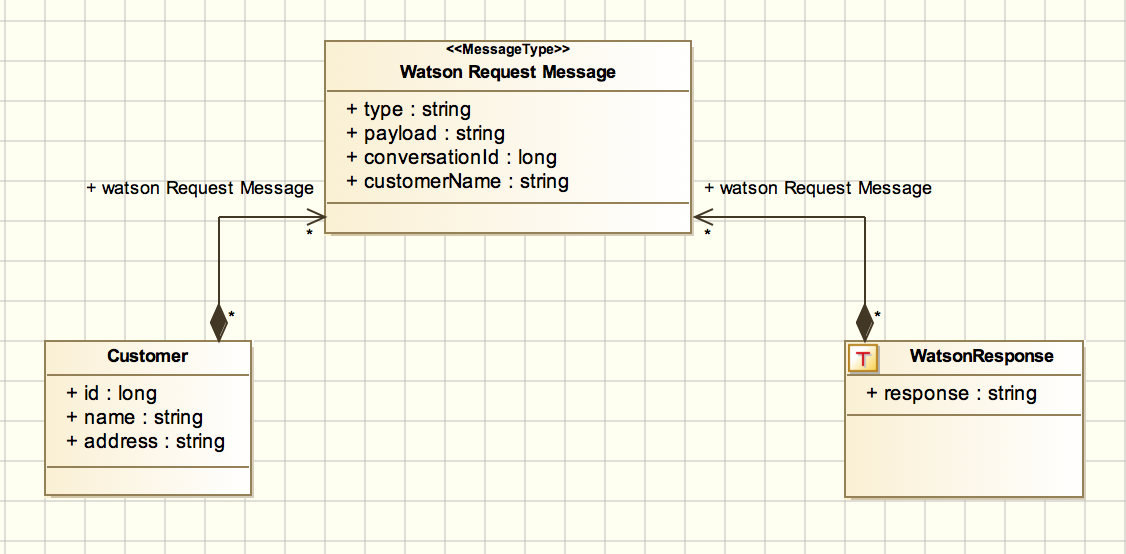
Google Maps expune utilizatorului informatii cartografice despre locatia pe care doreste sa o viziteze din momentul in care aceasta este selectata. Tot prin intermediul acestui serviciu sunt evidentiate locatiile de interes alese in functie de preferintele (subcategoriile) specificate prin marcatori prezenti pe harta.

**4. DarkSky.net**

Serviciul este folosit pentru a informa utilizatorul despre cum va fi vremea in orasul si in perioada in care vrea el sa calatoreasca.

Acesta va intoarce o lista cu prognoza meteo pentru fiecare zi din perioada aleasa de catre utilizator.

1. **Implementarea serviciilor**
   1. **Watson**



public ConversationDTO sendMessage(ConversationDTO conversationDTO) throws ParseException {

String conversationId = conversationDTO.getConversationId();

if (StringUtils.isEmpty(conversationId)) {

String uuid = UUID.randomUUID().toString();

conversationDTO.setConversationId(uuid);

conversationId = uuid;

contexts.put(uuid, new HashMap<String, Object>() {

{

put("conversation\_id", uuid);

}

});

}

Map<String, Object> context = contexts.get(conversationId);

ConversationDTO response = new ConversationDTO();

MessageRequest messageRequest;

if (contexts.get(conversationId) != null) {

messageRequest = new MessageRequest.Builder()

.inputText(message)

.context(context)

.build();

} else {

messageRequest = new MessageRequest.Builder()

.inputText(message)

.build();

}

MessageResponse messageResponse = service.message(workspace, messageRequest).execute();

// parsare messageResponse si trimitere mesaj catre utilizator

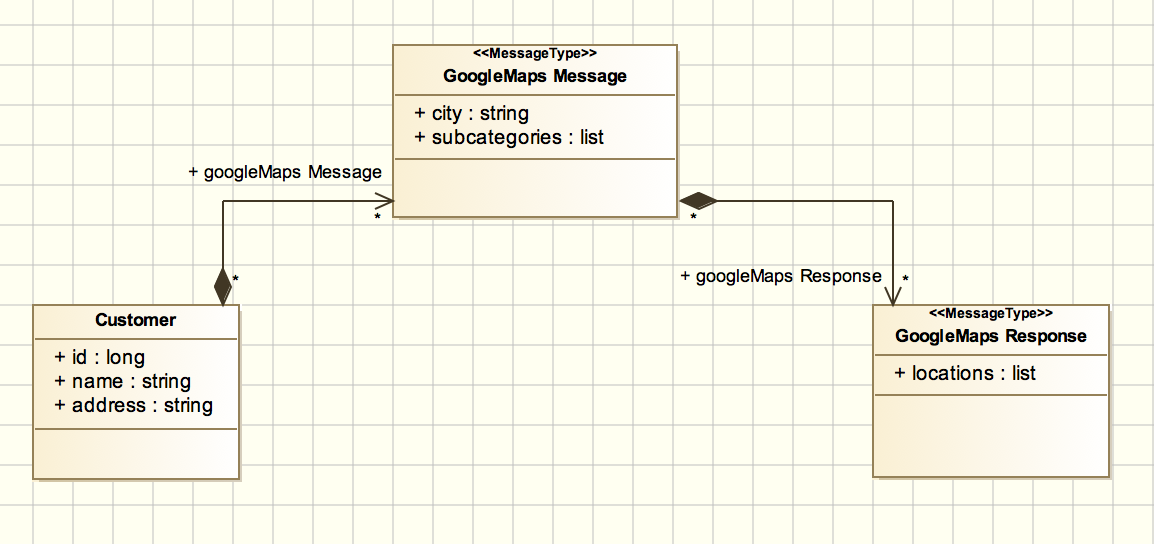
….

}

Interfata Android/Web cu Watson:

|  |  |
| --- | --- |
| **Android** | **Web** |
|  | screen6.png  screen7.png |

* 1. **Google Maps**



public List<Place> searchPlaces(String city, String subcategory) {

GooglePlaces googlePlaces = new GooglePlaces(key);

GooglePlaces.Param type = GooglePlaces.Param.name("type");

type.value(subcategory);

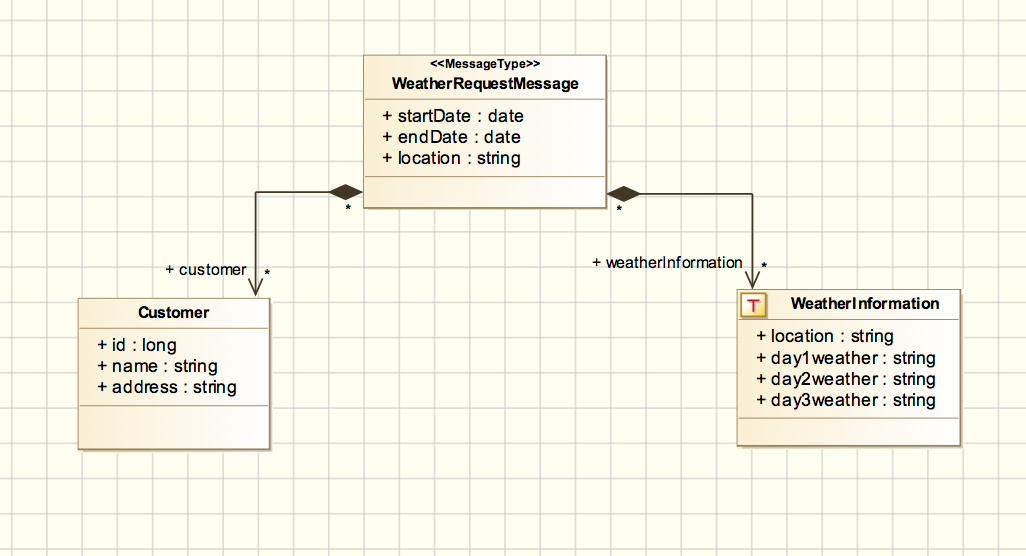
return googlePlaces.getPlacesByQuery(subcategory + " in " + city, NUMBER\_OF\_LOCATIONS);

}

Interfata Android/Web cu serviciul de Google Maps:

|  |  |
| --- | --- |
| **Android** | **Web** |
|  | screen3.png |

* 1. **Weather:**



while (startTime + offset < endTime + day) {

offset += day;

DsTimeMachineRequest request = DsTimeMachineRequest.builder()

.latitude(String.valueOf(location.getLatitude()))

.longitude(String.valueOf(location.getLongitude()))

.time(startTime + offset)

.excludeBlock(DsBlock.ALERTS, DsBlock.MINUTELY, DsBlock.HOURLY)

.unit(DsUnit.SI)

.build();

DsResponse response = client.sendTimeMachineRequest(request);

List<DsDataPoint> dayForecast = response.daily().data();

…. // extragere informatii meteo din aceasta lista si trimitere catre utilizator

}

Interfata Android/Web cu serviciul de weather:

|  |  |
| --- | --- |
| **Android** | **Web** |
|  | screen4.png |