Домашна работа 2.

1.Архитектура

1.1. Вовед

Нашиот тим одлучи дека ќе ги следи стапките по кои се движевме низ презентациите, односно ќе ги следи точните упатства кои ни беа препорачани. Па затоа ќе изработиме архитектури од трите дадени погледи во деловите 1.2 1.3 и 1.4 Во делот 1.2 каде што ќе ја опфатиме концептуалната архитектура ќе започнеме со барањата кои ги наведовме во домашна број еден вадејќи ги клучните концепти и правејќи категоризација по што ќе започнеме со архитектурата од екстерните системи и стејкхолдерите. Во 1.3 ќе направиме архитектурен модел во извршен поглед .На крајот во делот 1.4 ќе почнеме со тежинска табела во која се избира соодветна технологија и ќе ја прикажеме нашата финална (имплементациска) архитектура со детали како и на кој начин планираме да ја реализираме дадената архитектура.

1.2. Концептуална архитектура

На почетокот од првичните барања ги подвлековме клучните концепти за да можеме да направиме категоризација.

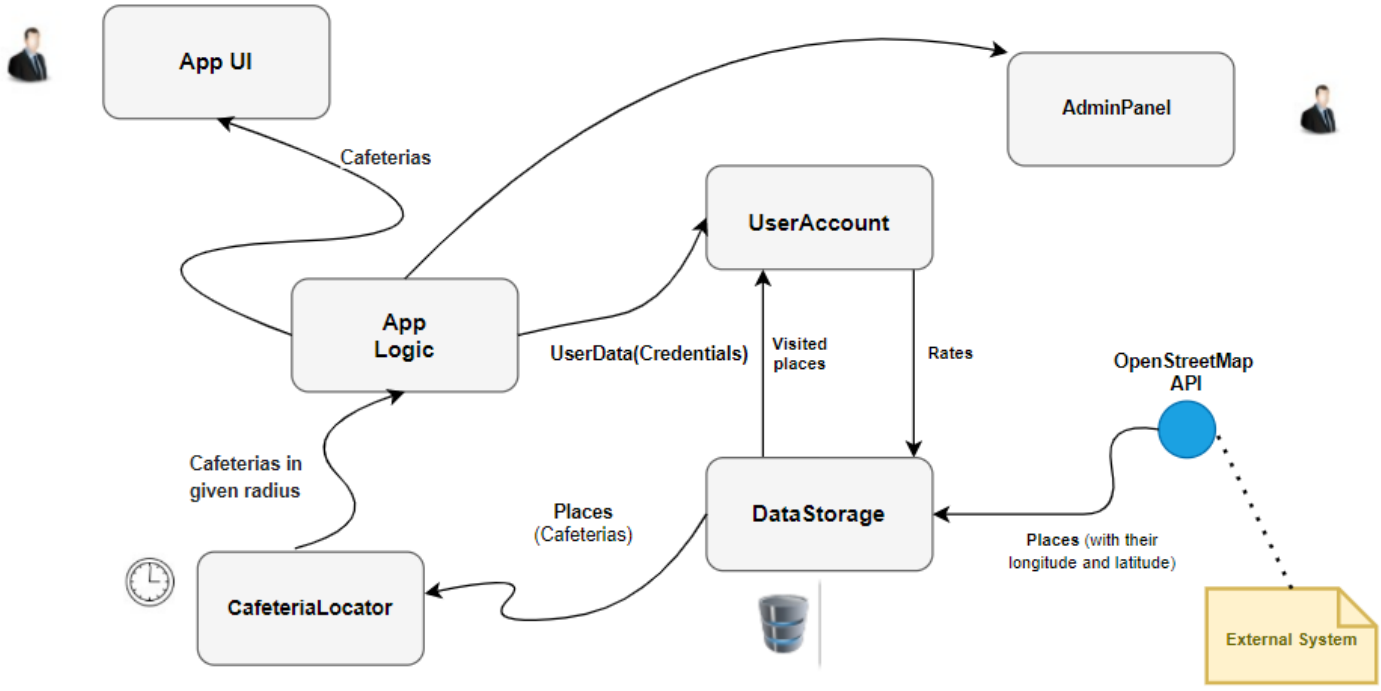
1. Web апликацијата ќе овозможува најава преку постоечка емаил адреса на корисникот.
2. Web апликацијата ќе овозможува лоцирање на објекти во избран радиус.
3. Web апликацијата ќе овозможува приказ на рута за пристигање до избраниот објект.
4.Web апликацијата ќе овозможува преглед на оценките кои се однесуваат за даден објект.
5.Web апликацијата ќе овозможува оставање на оценка на даден објект.
6.Web апликацијата ќе нуди можност споделување на локацијата.
7.Web апликацијата ќе овозможува преглед на попустите во одредени објекти кои ги нудат.
8.Web апликацијата ќе нуди можност за избор на радиус во кој ќе бидат означени објектите.
9.Web апликацијата ќе нуди филтер на објектите врз основа на нештата кои се нудат.
10.Web апликацијата ќе овозможува приказ на времето потребно да се стигне до дестинацијата.
11.Web апликацијата ќе нуди можност за поставување слика при оценување на даден објект.
12.Web апликацијата ќе дава можност за приказ на најдобро оценетите објекти.
13.Web апликацијата ќе нуди можност за размена на локација помеѓу корисници.
14.Web апликацијата ќе престане да ја користи локацијата на корисникот во моментот кога ќе го исклучи пребарувачот.
15.Web апликацијата ќе дава можност за зачувување на омилени места.
16.Web апликацијата ќе ја одбие секоја оценка ако корисникот никогаш не избрал рута да стигне до објектот кој го оценува .
17.Web апликацијата ќе нуди интерактивна мапа на која корисникот може да го следи своето движење.

18.Web апликацијата ќе нуди можност за прнаоѓање места само на територијата на Скопје.
19.Web апликацијата ќе нуди можност на пронаоѓање објекти кои нудат храна,пијалоци и учење.

System	Funcionality	Data	Stakeholder	Abstract Class
OpenStreetMap	најава	радиус	корисник	оценка
	лоцирање	локација на објект		попусти
	приказ на рута	територија на Скопје		омилени
	оставање оценка	оставени оценки		време
	споделување локација			дестинација
	избор на радиус			интерактивна мапа
	филтер			
	следење на движењето			

Табела 1.0. Категоризација на клучни концепти

Како што може да се забележи имаме еден надворешен систем и само еден стејкхолдер кој е опфатен во барањата.



Слика 1.0 Концептурална архитектура

App UI:

-Презентациски слој
-Менаџирање на output-от
-HTTPS конекција со AppLogic
-Рендерирање на мапа.

AppLogic:

-Пресметување на координати
-Наоѓање на објекти даден радиус
-Конекција со база
-Автентикација на корисниците

DataStorage:

-Податоци за името, адресата, типот на кафетеријата заедно со координатите .
-Податоци за најава
-Оценки за кафетеријата

UserAccount:

-Менаџирање со податоците на корисникот .
-Дел од AppLogic.

AdminPanel:

-UI за интерфејс на корисниците со привилегии.

OpenStreetMaps:

Надворешен систем од кој ги преземавме податоците за објектите како и мапата на UI.

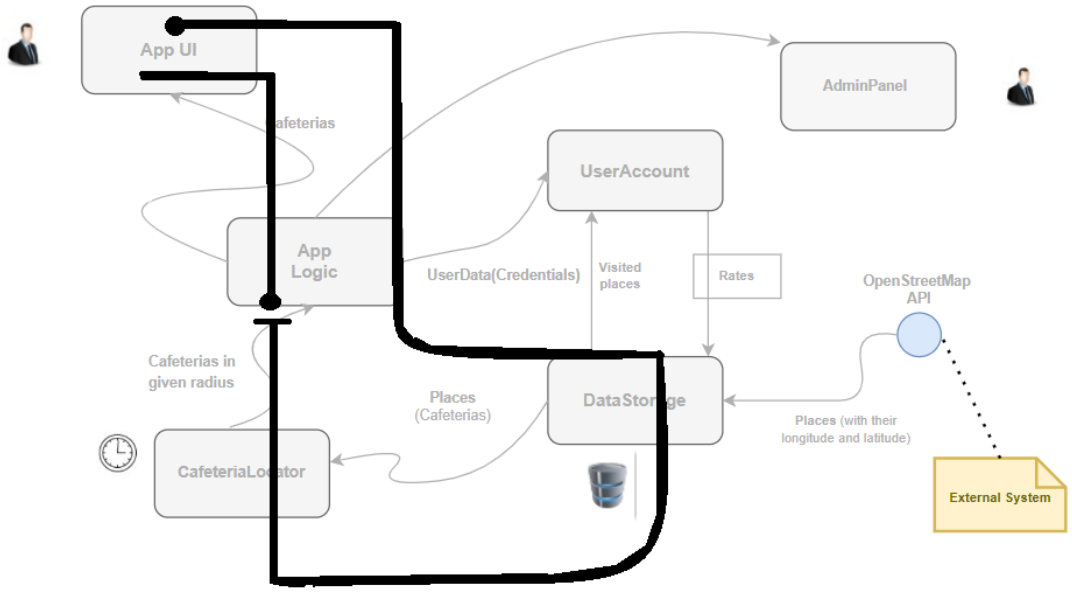
CafeteriaLocator:

-Наоѓање на објекти во даден радиус .
-Трансформација на податоци ако е потребно

Заради порано утврдување на грешки и потребата од Use Case мапа во оваа архитектура се одлучивме за методот со personna. Во овој метод ќе измислиме сценарио во кое може да се најде нашиот фиктивен лик Марија додека е на факултет и има празнина 1 час меѓу предавањата. Таа би сакала место каде ќе може да учи, а истовремено да пие кафе.

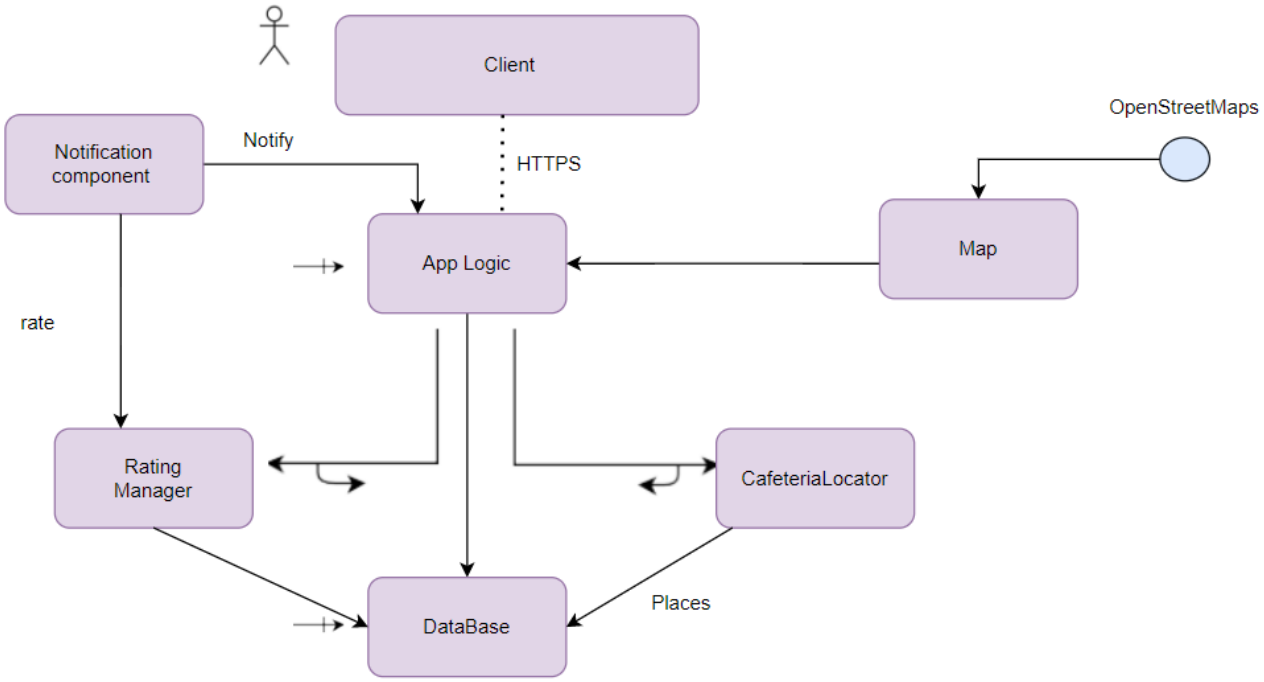
Personna: Марија, 22години, студент на ФИНКИ

Марија е студент на ФИНКИ која во моментот станува во еден од домовите, кои не се блиску до факултетот. Со оглед на тоа дека следното предавање и е само за еден час таа нема време да оди до домот и да се врати на факултет. Затоа Марија одлучува да ја користи нашата апликација за да најде објект близу неа каде може да пие топол пијалок, а истовремено да учи.



Слика 1.1. Use Case Map на концептуалната архитектура

1.3. Извршна архитектура



Слика 1.2. Извршна архитектура на Findify

Client:

-Апликациски интерфејс кој се презентира на клиентот
-Client sender и handler за requests.

Notification component:

-Ќе го извести системот кога ќе има внес на нова оценка за објектот.

RatingManager:

-Оваа компонента е поврзана со AppLogic преку CallBack врска.
--Внес на оценка
-CRUD функционалности.

AppLogic:

-Скоро сите функционалности
-Автентикација
-Трансформирање на податоци
-Пресметка на радиус
-Пресметка на објекти во близина
-Синхрона врска со базата
-Функција како MVC контролер

Map:

-Интерактивна мапа превземена од OpenStreetMap.
-Сервис на апликацијата.

DataBase:

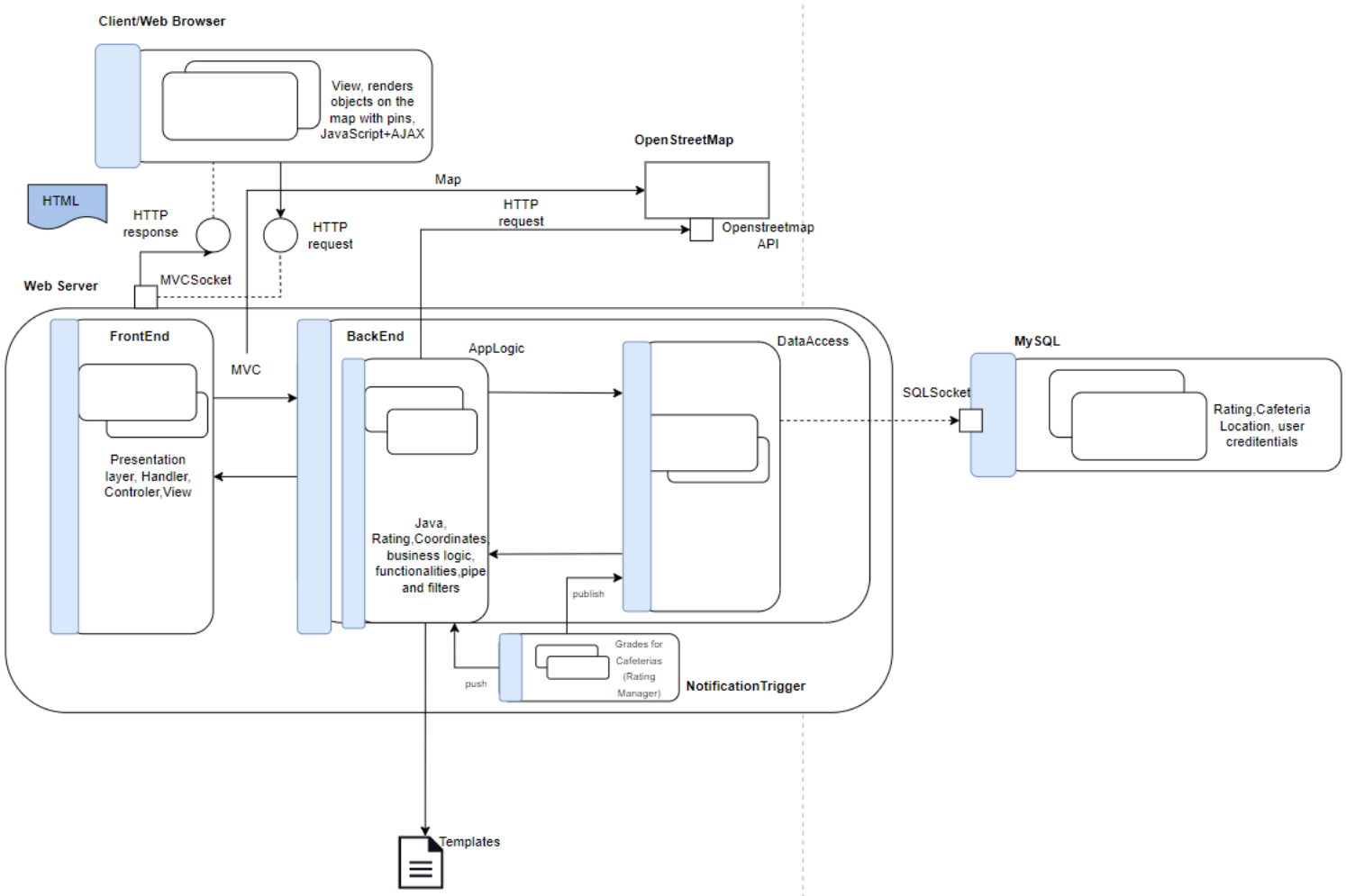
-Податоци за името, адресата, типот на кафетеријата заедно со координатите .
-Податоци за најава
-Оценки за кафетеријата
-CRUD фунцкионалности

1.4. Имплементациска архитектура (финална, хибридна)

Во оваа архитектура на почетокот искористивме тежинска табела со која што ги избравме соодветните технологии со оглед на знаењето на членовите од тимот и функционалностите кои ги зададовме во барањата.

Criteria (Бројот во ќелијата соодветствува со наведените барања во 1.2. Концептуална архитектура)	**weight** (тежината се движи од 1 до 5)	**Java + Spring**		**C#**		**HTML/CSS + JavaScript**		**React.js**	
		rating	**weighted score**	**rating**	**weighted score**	**rating**	**weighted score**	**rating**	**weighted score**
1	2	80%	1.6	60%	1.2	90%	1.8	50%	1
2	5	70%	3.5	50%	2.5	90%	4.5	50%	2.5
3	3	70%	2.1	55%	1.65	80%	2.4	60%	1.8
4	3	76%	2.28	60%	1.8	70%	2.1	60%	1.8
5	3	80%	2.4	50%	1.5	80%	2.4	50%	1.5
6	1	68%	0.68	70%	0.7	70%	0.7	60%	0.6
7	2	70%	1.4	60%	1.2	90%	1.8	50%	1
8	4	80%	3.2	50%	2	80%	3.2	60%	2.4
9	3	90%	2.7	70%	2.1	80%	2.4	60%	1.8
10	2	80%	1.6	60%	1.2	70%	1.4	50%	1
11	1	70%	0.7	70%	0.7	70%	0.7	60%	0.6
12	2	70%	1.4	50%	1	80%	1.6	60%	1.2
13	2	60%	1.2	40%	0.8	70%	1.4	50%	1
14	2	70%	1.4	30%	0.6	75%	1.5	50%	1
15	3	80%	2.4	50%	1.5	80%	2.4	50%	1.5
16	1	70%	0.7	60%	0.6	90%	0.9	60%	0.6
17	1	85%	0.85	70%	0.7	95%	0.95	60%	0.6
18	5	90%	4.5	60%	3	70%	3.5	50%	2.5
19	5	90%	4.5	60%	3	90%	4.5	40%	2
Totals	50	76%	39.11	57%	27.75	80%	40.15	54%	26.4
	Изберена технологија	**Java + Spring**				**HTML/CSS**			
	Rating: Excellent ★★★★★(100%); Good ★★★★(75%); satisfactory ★★★ (50%); mediocre ★★ (25%); poor ★ (0%)								

Табела 1.1. Тежинска табела за избор на технологија при имплементација



Слика 1.3. Имплементациска (хибридна) архитектура .

Client/Web Browser:

-Пребарувач кој користи униформен апликациски протокол HTTPs
-Рендерира поглед на корисникот
-На клиентска страна како pushed функционалност ќе биде исцртувањето на мапата со вратени pins од серверот заради loose coupling а воедно и ќе го намалиме оптеретувањето на серверот.(Separation of concerns)
-праќа HTTPs барање и чека одговор (синхрона комуникација)

WebServer:

Овој дел заради поголема прегледност го поделивме на Frontend и Backend

- Frontend технологија која ја користиме HTML/CSS со JavaScript и тука се наоѓа првиот слој MVC.

-Backend технологија Java+Spring, тука се наоѓа вториот слој а тоа е AppLogic.

MVC:

-Првиот слој во архитектурата кој е од делот Frontend
-Задолжен за комуникација со клиентите и известувања.
-Задолжен за препаќање на барањата од клиентите до серверот (vice versa).
-Одговорен за креирање на погледите кои ги нуди интерфејсот.
-Имплементација на интерактивна мапа преземена од OpenStreetMap

AppLogic:

-Вториот слој во архитектурата
-Задолжен за комуникација и update со NotificationTrigger, односно кога корисникот ќе внесе оценка тоа ќе иницира промена на просечната оценка на објектот па овој слој е задолжен за update и примање на нотификациите од оваа компонента.
-Може да земе и модифицира податоци од базата
-Ги подготвува податоците за првиот слој да ги испрати на корисникот.

NotificationTrigger:

-Одговорен за пренос (push) на настанот (внесена оценка) до AppLogic.
-Користи нотификациска компонента API за креирање нотификација во системот.
-Самата компонента ќе биде иницираната од секој нов настан односно внес.
-Има комуникација и со самата база на податоци каде ги праќа внесените оценки.

DataAccess

-Користи CRUD операции за податоците од базата.

DataBase

-PostgreSQL која ќе ја користиме за чување на конзистентните податоци потребни за имплементирање на функционалностите кои се опишани во барањата.

Team: Findify

FINDIFY

Navigation App