# Spletno programiranje, projektna naloga

## Gašper Andrejc, 63130002 2015/2016

Travel e-xpense

### https://travel-e-xpense.herokuapp.com/

## Opis, namen in izvedba

Glavni namen aplikacije »Travel e-xpense« je informatizirati proces in birokratske zahteve, ki nastanejo pri službenih potovanjih. Cilj je minimalizirati le-te in celoten proces, ki je sedaj več kot nadležen, narediti neopazen tako za uslužbence in managment kot tudi za računovodstvo, ki je končni prejemnik vsakega naloga. Vse od prvotne zahteve zaposlenega za neko potovanje, do potrditve s strani managmenta in na koncu do zaključka naloga s strani računovodstva se z aplikacijo informatizira. Prav tako se vsemi omenjenimi akterji nudi tudi splošen organiziran in enostaven pregled nad procesom (pretekli nalogi, vsi aktivni, ...) in komunikacijo med njimi. Na primer, če računovodstvo opazi nepravilnost na zaključenem nalogu ima možnost enostavno in hitro stopiti v kontakt – vice versa. Naslednja pomembna funkcionalnost aplikacije je transparentnost – hitro se lahko vidi vse pretekle naloge, vsa zapravljena sredstva in povezane vsebine. Prav tako so bile vzete v račun pomebne malenkosti, kot so anonimnost med računovodstvom in uslužbencem, da ne pride do subjektivnih ocen, in ostale, ki dajejo aplikaciji posebno dodano vrednost.

Poleg osnovnih funkcij omenjenih zgoraj je sekundarni cilj aplikacije nuditi uslužbencu v tujini osnovno pomoč, za kar so kar v aplikacijo integrirane zunanje storitve, ko je na primer zemljevid z možnostjo filtriranja rezultatov za stvari, ki na poti pridejo najbolj prav (parkirišča, rent-a-car, javni potniški promet, ...). Pri managmentu in računovodstvu je ta sekundarni cilj osnovna statistika in pregled nad podatki, ki lahko izboljšajo nadaljnjo poslovanje.

**Ciljna publika** aplikacije so predvsem podjetja, ki so veliko, tako ali drugače, aktivna zunaj lastnega sedeža – bodisi po državi ali pa kar internacionalno. To še zdaleč niso le IT podjetja, ampak tudi takšna, ki mogoče niso najbolj dojemljiva za tehnologijo – zaradi tega je bila odločitev, da se načrtuje čim bolj intiutiven work-flow z jasnimi preglednimi meniji in akcijami, ki jih uporabnik glede na vlogo lahko izvede. Po pregledu vseh možnih procesov je tudi sama struktura strani bila narejena tako, da so stvari, ki bodo največkrat uporabljene postavljene v ospredje ali pa je vsaj način dostopa intiutiven. Tako je na primer kar na prvi strani za uslužbence pregled nad trenutnim nalogom, kjer se ga lahko poljubno ureja oz. dodaja storitve »on-the-fly« brez večjega napora. Enako intutivino je narejen tudi »dashboard« za računovodstvo, kjer je vbistvu izbira naloga, ki se ga bo naslednjega pregledalo takšna, da se klikne na puščico, ki nalog »povleče« na glavno urejevalno površino (kot da bi papirnat nalog povlekli predse).

## Zemljevid strani

Zaradi malo strani (večina funkcionalnosti je narejena s »pop-upi«) sem se odločil, da se uporabi linearna organizacija spletišča, ki je zaradi zgoraj omenje funkcionalnosti precej enostavna. Glej prilogo website\_map.pdf

## Podpora vseh priljubljenih brskalnikov

Testirano na:

* Chrome 46.0.2490.86 m
* Microsoft Edge
* Firefox 41.0.1

## Kratek pregled zame zanimivih stvari

* QR code reader je definitivno integracija, ki mi je bila najbolj zanimiva
* Google maps z integracijo najbolj pogostih service-ov, ki bi jih nekdo na poti potreboval
* Veliko stvari rešenih z »iz nič narejenimi« pop-upi, kar mi je všeč ker se mi zdi veliko statičnih strani linkanih s <a> elementi staromodno in nefunkcionalno
* Veliko uporabljen javascript za izboljšanje uporabniške izkušnje, npr. pri potrditvi naloga (animacija), pri filtriranju rezultatov s ključnimi besedami
* Prikaz sporočil z animacijo
* Ideja »povezanih stvari« pri sporočilih, kjer lahko hitro ugotoviš na kaj točno se sporočilo navezuje
  + Enako pri 'Mojih nalogih', kjer so integrirana še povezana sporočila

## TODOs

* Prikaz nalogov in zahtevkov je trenutno enak ne glede na vlog oporabnika – ker se bo vloga preverjala v zalednem sistemu sem to pustil za kasneje – informacije in akcije, ki jih lahko uporabnik naredi na nekem nalogu/zahtevku se bodo razlikovale glede na vloge
* Stvari kot so dodajanje in pošiljanje v arhiv mora seveda biti podprto v zalednem sistemu z bazo
* Osvežitev čakajočih nalogov pri /finance/ mora biti implementirano v zalednem z bazo
* Implementacija dejanskih sporočil med uporabniki v zalednem sistemu (+ algoritem za povezovanje med nalogi, zahtevki in sporočili)

# Faza 2:

**Namestitev aplikacije:**

**P**otrebujete:

* Wildfly aplikacijski strežnik verzije 8 ali več, lahko tudi jboss 7
* jdk 6+
* MySql podatkovno bazo
* Maven verzije 3 (lahko tudi 2, ker je 3 backward compatible)

Ko imate wildfly naložen odprite wildfly/standalone/configuration/standalone.xml ter vanj skopirajte naslednji datasource:

<datasource jndi-name="java:jboss/datasources/sp\_vaje" pool-name="sp\_vaje" enabled="true" use-java-context="true">

<connection-url>jdbc:mysql://localhost:3306/travel-expense</connection-url>

<driver>mysql</driver>

<security>

<user-name>root</user-name>

<password>root</password>

</security>

<validation>

<validate-on-match>false</validate-on-match>

<background-validation>false</background-validation>

</validation>

<statement>

<prepared-statement-cache-size>0</prepared-statement-cache-size>

</statement>

</datasource>

**Nastavitve v sami aplikaciji**

Odprite glavni pom.xml ter v <jboss.deploy.dir> nastavite pot do svojega strežnika /deployments mep, tja not se namreč skopira .ear datoteka po maven instalaciji aplikacije.

**Buildanje aplikacije**

V konzoli se premaknite v glavni pom.xml ter poženite ukaz 'mvn clean package' ali 'mvn clean install'. To aplikacijo skopira v zgoraj omenjeno datoteko.

**Zagon aplikacije**

Aplikacijski strežnik poženete tako, da se v konzoli premaknete v /wildfly/bin in poženete standalone.bat/.sh.

## Opis faze 2

Za backend sem se odločil za JavoEE 7 z JavaFaces JSF frontend implementacijo. Razlog za to je, ker java ponuja veliko dobro testiranjih in vsesplošno uporabljenih knjižnic za delo s podatkovo bazo in podobnimi stvarmi, ki jih lahko vzamemo za samoumevne in se nam z njimi ni potrebno preveč ukvarjati.

Podatkovna baza je MySql, prav lahko pa bi bila tudi Postgres ali katera druga, saj sem Entity model implementiral z JPA ki je neodvisen od podatkovne baze zadaj.

**Cachiranje**

Uporabil sem cachiranje na nivoju backenda. Prvi request na bazo poleg tega, da vrne podatke hrani tudi v HashMap v beanu z application scope, kar pomeni, da se cache ohranja skozi celotno aplikacijo. Vsak naslednji request najprej pogleda v hashmap in vzame iz tam, če pa ga tam ne najde pa dostopa do bazo. Tako je ozko grlo, ki ga predstavlja dostop do baze kar se da malo uporabljeno. Lahko bi naredil tudi cachiranje na frontendu ali pa dvojno cachiranje v backendu a se zaradi malega števila podatkov s tem nisem ukvarjal.

**Varnost**

Za varovanje sem uporabil Javin standardni access-control-list skupaj s spletnimi filtri, ki jih ponuja. Tako se strani strežejo samo uporabnikom s pravicami. Npr celotna pot /user/\* je dovoljena samo uporabniku s to funkcijo.

**Logiranje na 2 nivojih**

Uporabil sem logiranje na noviju strežnika, kar pomeni logiranje exceptionov in flowa kot takega (kdaj se cachira, kdaj se cache pobriše, kdaj je kak ključen postopek zaključen ipd.) in pa tudi logiranje samega potega aplikacije z vodika uporabnika. Torej vse pomembne akcije, ki morajo biti trackane so logirane in hranjene v bazo, admin pa lahko do njih dostopa enostavno preko URLja ki podpira HTML izpis direktno iz baze (<http://localhost:8080/logging> in pa <http://localhost:8080/logging?type=4> kjer je type opcijski parameter, in sicer po naslednjem vrstnem redu:

USER\_ZAHTEVEK("user\_zht"),

USER\_NALOG("user\_nlg"),

MANAGMENT\_ZAHTEVEK("mngmnt\_zht"),

MANAGMENT\_NALOG("mngmt\_nlg"),

FINANCE("finance");

)

Da je to logiranje čim manj zahtevno za podatkovno bazo je narejeno tako, da se vse logiranje aplikacijskega flowa logira v cache, predno pa se aplikacija začne ustavljati pa se vsi logi persistirajo v bazo. Prav tako se persistiranje začne pred vsakim zahtevkom za vpogled v loge, da se lahko auditing izvede s svežimi podatki.

**Selenium testi**

Narejeni so bili 3 selenium testi, vsak za en flow uporabnika – driver, managment in user.