

»Mladi za napredek Maribora 2019«  
36. srečanje

# **SISTEM ZA PRIJAVO NA ŠPORTNE DNEVE IN OBVEZNE IZBIRNE VSEBINE**

**RAČUNALNIŠTVO**

*INOVACIJSKI PREDLOG*

Avtor: MIHA FRANGEŽ

Prostor za nalepko

Mentor: ALEŠ BEZJAK, BOJAN SKOK

Šola: SREDNJA ELEKTRO-RAČUNALNIŠKA ŠOLA MARIBOR

Število točk: 158

Mesto: 1

Priznanje: srebrno

Maribor, februar 2019

»Mladi za napredek Maribora 2019«  
36. srečanje

# **SISTEM ZA PRIJAVO NA ŠPORTNE DNEVE IN OBVEZNE IZBIRNE VSEBINE**

**RAČUNALNIŠTVO**

*INOVACIJSKI PREDLOG*

Prostor za nalepko

Maribor, februar 2019

# KAZALO VSEBINE

Povzetek.....	5
1 OPIS PROBLEMA.....	6
1.1 Cilj.....	6
1.2 Pregled obstoječih rešitev.....	6
2 TEHNIČNA IZVEDBA.....	9
2.1 Oblikovanje vmesnika.....	9
2.2 Podatkovni model.....	11
2.3 Programski jeziki in okolje.....	12
2.4 Baza uporabnikov.....	12
2.5 Gostovanje.....	15
2.6 Varnost in zasebnost.....	17
3 UPORABA APLIKACIJE.....	18
4 ANALIZA IZDELKA.....	20
5 DRUŽBENA ODGOVORNOST.....	21
6 ZAKLJUČEK.....	22
7 Zahvale.....	23
7.1 Uporaba odprte kode.....	23
8 VIRI IN LITERATURA.....	24

## KAZALO SLIK

Slika 1: Prototip uporabniškega vmesnika na papirju .....	9
Slika 2: Statični prototip domače strani.....	10
Slika 3: Statični prototip prijavnega vmesnika .....	10
Slika 4: Relacijska shema podatkovne baze .....	11
Slika 5: Izvoz podatkov iz eAsistenta .....	13
Slika 6: Uvoz podatkov iz preglednice .....	14
Slika 7: Vmesnik za urejanje dogodka .....	18
Slika 8: E-poštno sporočilo, ki ga prejmejo dijaki .....	19
Slika 9: Pregled prijavljenih dijakov .....	19

## **POVZETEK**

V srednjih in osnovnih šolah se vsako leto srečujemo s prijavi na razne šolske dejavnosti in športne dneve. Pri teh prijavi vedno pride do zmede z zbiranjem prijavi na razne aktivnosti, še posebej, če je na izbiro več dejavnosti v enem dnevu. Prav tako se lahko na nekatere dejavnosti prijaviš samo do zapolnitve mest.

Po svojih izkušnjah in po pogovorih s profesorji sem prišel do vprašanja: zakaj tega ne bi rešili elektronsko? Tako sem prišel do ideje, da bi naredil sistem, s katerim bi učitelji elektronsko vnesli aktivnosti v sistem, učenci pa bi dobili obvestilo, da se morajo na dejavnosti prijaviti preko spletnega vmesnika.

Naš cilj je ustvariti sistem, ki bo dijakom olajšal prijavo k šolskim dejavnostim in razvrstitev po skupinah, profesorjem pa omogočil boljši pregled nad udeleženi. Istočasno pa bomo poskusili ohraniti sistem čim bolj fleksibilen, da bo lahko uporaben tudi v drugih situacijah.

# 1 OPIS PROBLEMA

Problem, ki ga v naši nalogi želimo rešiti, je na prvi pogled precej enostaven. Na šoli se vsako leto izvajajo mnogi dogodki, kot so športni dnevi, obvezne izbirne vsebine in tekmovanja, na katere se morajo dijaki prijaviti.

Na večini šol, vključno z našo, se v ta namen še vedno uporabljajo enostavne preglednice na papirju, na katerega se dijaki vpisujejo ročno pri profesorjih oz. z okrožnicami, ki jih dežurni dijaki nesejo od razreda do razreda.

Ta sistem se nam zdi zelo neučinkovit, zato ga želimo zamenjati z računalniško rešitvijo v obliki spletne aplikacije, ki bi profesorjem omogočala razpis dogodka (npr. športnega dneva) in povabilo dijakov za prijavo nanj. Ker imajo mnogi dogodki tudi možnost izbire skupine (npr. smučanje, drsanje ali nogomet na športnem dnevu), mora naša aplikacija omogočati tudi to. Omogočati mora tudi dodajanje omejitev in roka prijave, podrobne opise dogodkov in obveščanje po e-pošti.

## 1.1 Cilj

V fazi načrtovanja smo si zadali določene cilje oz. zahteve, ki jim mora končni izdelek zadovoljiti, da ga štejemo kot uspeh:

1. dijakom mora biti uporaba uporabniškega vmesnika enostavna in jasna brez predhodne razlage;
2. aplikacija mora biti integrirana z obstoječo šolsko infrastrukturo (torej mora uporabljati obstoječe podatke o dijakih ter prijavo z gesli, ki jim že imajo vsi dijaki);
3. aplikacija mora delovati na čim večjem obsegu naprav, ki jih uporabljajo dijaki;
4. aplikacija mora omogočati minimalno vse, kar omogoča "analogni" pristop, torej:
  - o prijave se lahko naknadno spreminjajo s strani dijakov in profesorjev,
  - o dijaki se lahko za skupino odločijo tudi glede na to, kateri sošolci so že prijavljeni,
  - o profesorji lahko po potrebi spreminjajo prijave vseh dijakov,
  - o profesorji lahko po potrebi prekoračijo omejitve prijav;
5. aplikacija mora pri normalnem poteku dela delovati brez potrebe za tehnično podporo (to vključuje situacije, kot so prepisi dijakov, menjava šolskega leta ...).

## 1.2 Pregled obstoječih rešitev

Pred začetkom razvoja smo pregledali še nekaj obstoječih rešitev ter se pogovorili z njihovimi uporabniki.

Začeli smo seveda na naši šoli, kjer se še vedno uporablja "analogni" pristop – preglednica na papirju. Čeprav omogoča večjo fleksibilnost kot katera koli izmed računalniških rešitev, smo po pogovoru s profesorji našli veliko pomanjkljivosti. V primerih, ko profesorji zbirajo prijave sami, največji problem povzročajo manjkajoči dijaki. Dijak, ki ga pri uri športne vzgoje pred športnim dnevom ni, si ne more izbrati skupine. Ko se vrne, mora poiskati profesorja izven pouka ter ga prositi, če se lahko prijavi.

Drug problem tega pristopa pa je spreminjanje prijav. Dijaki se pogosto v tistem trenutku napačno odločijo ali pa iz kakršnih koli drugih razlogov kasneje želijo spremeniti prijavo. V tej situaciji morajo tudi oni poiskati profesorja in ga prositi, da prijavo spremeni.

Ta problema so na eni izmed šol, na katerih smo se pozanimali, poskusili rešiti z eno veliko tabelo, izobešeno na oglasni deski, na katero so se dijaki lahko vpisovali sami. Čeprav so s tem olajšali delo profesorjem kot tudi dijakom, pa so hitro ugotovili, da so nekateri dijaki začeli sistem zlorabljati in spreminjati prijave drugih dijakov.

Digitalna rešitev lahko reši oba problema, zato so jo nekatere šole že poskusile uvesti. Prva izmed teh šol je papir nadomestila z deljeno Excel Online preglednico, v kateri so posamezne liste urejali predstavniki oddelkov. Samodejno vodenje različic je odvrčalo goljufanje, čeprav ga ni popolnoma preprečilo, delo pa je s profesorjev le preneslo na predstavnike oddelkov.

Na drugi šoli so se odločili za uporabo spletne učilnice *Moodle*. Večina dijakov jo je že uporabljala, zato je sama prijava na dogodke potekala precej gladko, do težav pa je prišlo, ko so profesorji želeli pregledati prijave. Pogledi, ki jih podpira vmesnik *Moodle*, enostavno niso omogočali vsega, kar so profesorji potrebovali.

Ena izmed šol pa je za prijavo razvila tudi lasten sistem. Veliko podrobnosti nam o njem niso znali povedati, saj z avtorjem aplikacije niso več v stiku. Aplikacija je bila precej osnovna po izgledu in delovanju – dijak se je prijavil na spletno stran, kjer je v spustnem seznamu lahko izbral eno izmed razpisanih skupin. Čeprav ga tamkajšnji profesorji uporabljajo raje kot navadne preglednice, pa s sistemom niso pretirano zadovoljni. Največji problem, ki smo ga našli tudi v sistemu *Moodle*, je, da ne omogoča omejitve vpisa po posameznih oddelkih. Rešitev, ki so jo uporabili na šoli s sistemom *Moodle*, je uporaba ločenega dogodka za vsak oddelek, kar pa je preglednost prijav še poslabšalo.

Dva druga sistema, ki rešujeta problem, podoben našemu, sta *Platforma SKOZ*<sup>1</sup> – projekt Gimnazije Vič – ter prijavni sistem projekta *RaST*<sup>2</sup> II. gimnazije Maribor. Oba sta nastala za podporo projektov, ki razpisujejo delavnice, tekmovanja in natečaje, na katere se dijaki lahko samostojno prijavijo. Uporabniška izkušnja obeh je precej podobna tej, ki jo želimo doseči mi, oba pa imata eno veliko pomanjkljivost, ki ju naredi neprimerne za naš problem – ne podpirata omejitve prijav glede na oddelek, kar je ena izmed naših ključnih zahtev.

Ogledali smo si tudi nekaj bolj splošnih rešitev za organizacijo dogodkov, a nobena izmed njih ni zadovoljila vseh zahtev. Omejiti smo se morali na odprto-kodne aplikacije, saj zaradi varstva podatkov ne moremo uporabiti aplikacije v oblaku, za integracijo s šolsko infrastrukturo pa je možnost modifikacije izvorne kode praktično nujna.

<sup>1</sup> <https://platforma.skoz.si>

<sup>2</sup> <http://prijava.projekt-rast.si>





## 2 TEHNIČNA IZVEDBA

Tehnična izvedba je potekala v treh fazah.

V prvi fazi smo pripravili del aplikacije, ki ga vidi uporabnik (t. i. frontend). To je vključevalo sam izgled aplikacije, kot tudi potek interakcije uporabnika z aplikacijo (t.i. user flow).

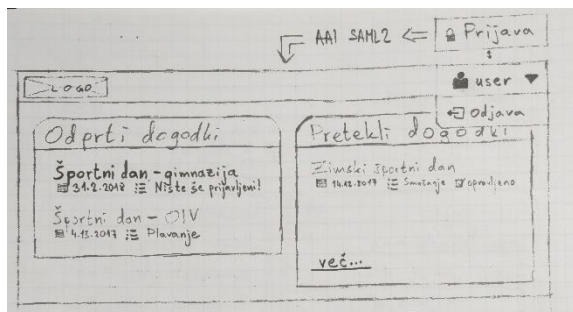
V drugi fazi smo načrtovali in implementirali ozadje aplikacije (t. i. backend), ki bo prej definirano uporabniško izkušnjo omogočal. Tukaj smo torej oblikovali podatkovne modele, implementirali vse operacije znotraj aplikacije ter vse to dinamično povezali z vmesnikom.

V tretji fazi pa smo aplikacijo namestili na strežnik in preizkusili v praksi. Med preizkušanjem smo sproti odpravljali napake ter dopolnjevali aplikacijo dokler ni zadostovala vsem zahtevam.

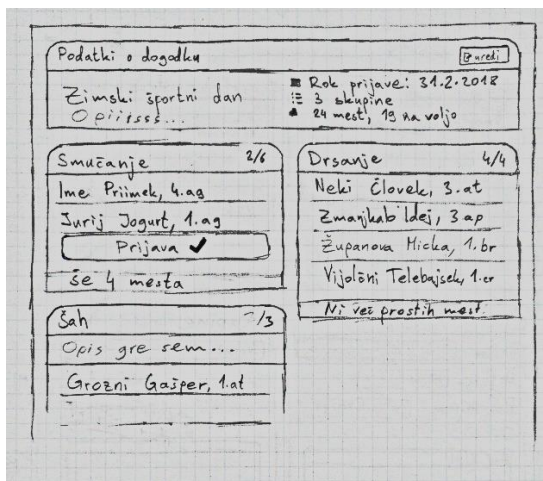
### 2.1 Oblikovanje vmesnika

Kot po navadi se je razvoj izgleda vmesnika začel na papirju. Pripravili smo skice vmesnika in jih pokazali nekaj dijakom, da bi ugotovili, če se na vmesniku znajdejo.

Precej hitro smo tako prišli do spodnjega izgleda, ki so ga vsi dijaki pravilno razumeli hitro in brez razlage.



Slika 1: Prototip uporabniškega vmesnika na papirju

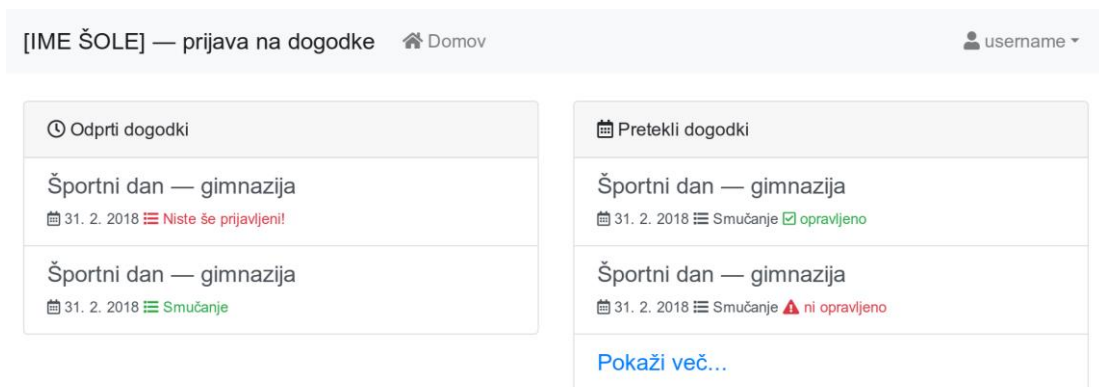


Ko je bil končen izgled vmesnika določen, smo se lotili statičnega prototipa. V tem smo zajeli vsa možna stanja aplikacije z izmišljenimi podatki.

Prototip smo pripravili v obliki spletne strani, kar nam je kasneje omogočilo veliko lažjo pretvorbo le-tega v končni, dinamični vmesnik.

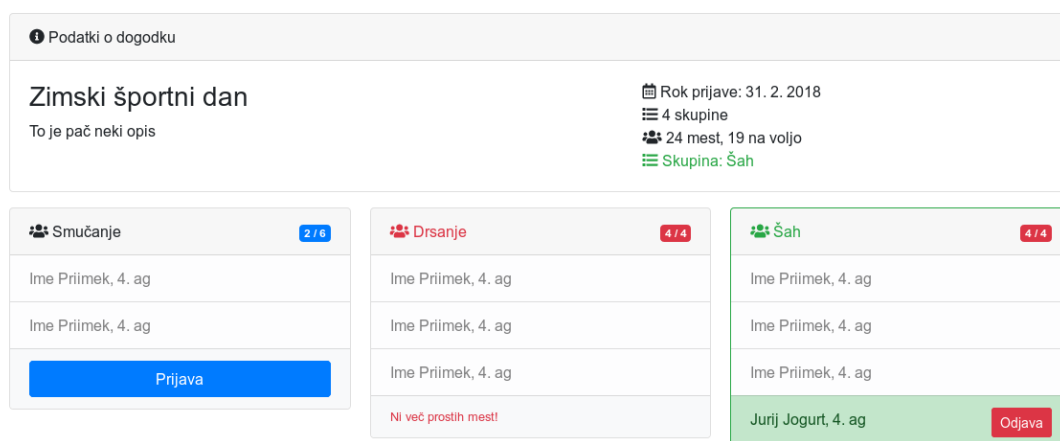
Za ogrodje vmesnika smo uporabili knjižnico *Bootstrap 4*, ki je razvoj precej pospešila, še pomembneje pa je z malo dodatnega dela omogočila zanesljivo delovanje vmesnika na širokem naboru naprav (glede na resolucijo, obliko zaslona, tip vmesnika in brskalnik).

Prva stran, ki jo uporabniki vidijo po vstopu v aplikacijo, je domača stran, ki jim omogoča pregled nad dogodki, na katere so povabljeni. Dogodki, za katere so prijave še vedno odprte (torej rok za prijavo še ni pretekel), so prikazani na vrhu (oz. na levi strani na širših zaslonih), pretekli pa spodaj/ob strani. Vsak dogodek je prikazan s kratkim povzetkom, ki vključuje naslov, datum dogodka in stanje prijave. Za pretekle dogodke smo v prototipu dodali še oznako prisotnosti, saj je bila na začetku predvidena tudi možnost evidence prisotnosti na dogodku, a v prvi različici tega nismo implementirali.



Slika 2: Statični prototip domače strani

S klikom na dogodek se uporabniku odpre pogled dogodka. Tukaj je dogodek predstavljen podrobneje, s celotnim opisom in seznamom skupin. Vsaka izmed skupin je predstavljena s svojo kartico, ki prikazuje stanje prijave in seznam prijavljenih dijakov iz istega oddelka. Če dijak skupine še ni izbral, se lahko prijavi v katero koli prosto skupino; če jo je že, pa se lahko po želji tudi odjavi in prijavi v drugo.



Slika 3: Statični prototip prijavnega vmesnika

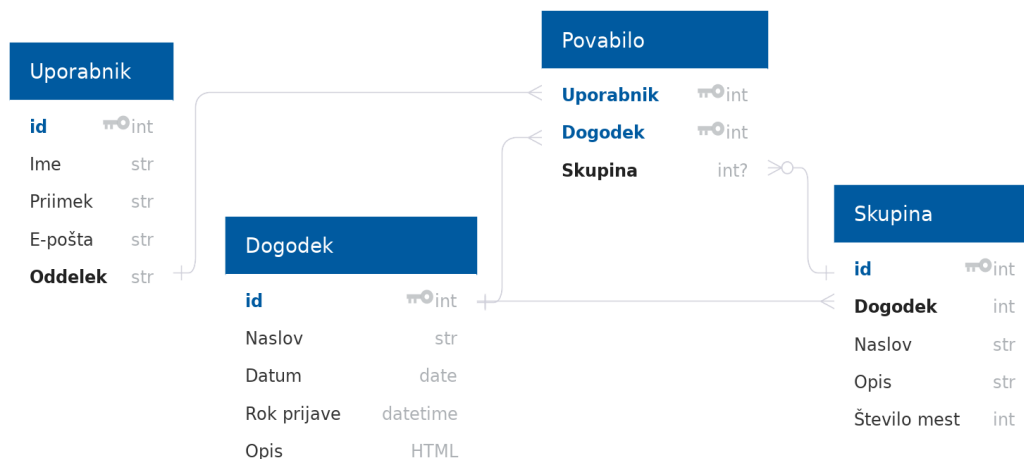
## 2.2 Podatkovni model

V aplikaciji imamo tri glavne entitete: uporabnike (dijake), dogodke in skupine. Najpomembnejša podatka uporabnikov sta e-poštni naslov, ki je enolični identifikator dijaka na šoli in je potreben za prijavo, ter oddelek, ki je potreben za omejitve dostopa in prijav.

Dogodki imajo poleg naslova in datuma še rok prijave, ki predstavlja točni čas, ko se prijave zaprejo, ter opis, ki smo ga implementirali kot polje HTML, da lahko v njem uporabljamo napredno oblikovanje in pripenjanje datotek (seveda s pomočjo WYSIWYG<sup>3</sup> urejevalnika).

Posamezne skupine imajo tudi svoje naslove in opise, definirajo pa tudi maksimalno število prijav, ki velja za vsak oddelek posebej.

Vsak dogodek ima lahko več skupin, vsak dijak pa se lahko pri posameznem dogodku prijavi v le eno skupino. To smo dosegli s povezavo skupine nazaj na pripadajoč dogodek ter vmesno tabelo (»Povabilo«), ki povezuje uporabnika na dogodek ter, ko se prijavi, v skupino.



Slika 4: Relacijska shema podatkovne baze

Uporabniku damo do dogodka dostop tako, da ustvarimo povabilo, ki ga povezuje na dogodek. Ko uporabnik izbere skupino, se ta zapiše v atribut ustreznega povabila.

<sup>3</sup> *What You See Is What You Get* – princip vizualnega urejevalnika, ki omogoča enostavno oblikovanje besedila

## 2.3 Programski jeziki in okolje

Za programski jezik, ki bo poganjal funkcionalen del aplikacije, smo izbrali *Python*. Hitrost, varnost in prenosnost so njegove glavne prednosti, glavni razlog za našo izbiro pa je bila izjemna izbira knjižnic in aplikacij, s katerimi lahko olajšamo razvoj.

Ogrodje spletne aplikacije (ang. *framework*) predstavlja sistem *Django*. Ta v en paket združi knjižnico za abstrakcijo podatkovne baze (ang. *Object Relational Mapping – ORM*), samodejno posodabljanje relacijske sheme (ang. *schema migration*), vmesnik za komunikacijo s spletnim strežnikom (ang. *Web Server Gateway Interface – WSGI*), vsebuje pa tudi lasten sistem predlog (ang. *templating engine*) in priročno orodje za upravljanje s podatki aplikacije: *django-admin*.

Še ena prednost sistema *Django* je podpora za vključevanje zunanjih modulov oz. aplikacij. To nam omogoča uporabo odprto-kodnih modulov, ki so že napisani, in tako poenostaviti razvoj.

*Python* okolje aplikacije smo upravljali z orodjem *pipenv*. To iz definicije zahtev aplikacije, ki jo shranimo zraven kode, na vsakem računalniku samodejno postavi identično okolje, v katerem bo aplikacija lahko delovala brez težav. Za konfiguracijo pa smo poskrbeli z uporabo spremenljivk okolja (ang. *environment variables*) in modula *django-environ*.

Kombinacija teh orodij nam omogoča implementacijo metodologije *Twelve-Factor App* (Wiggins A., 2017), ki bo naši aplikaciji omogočala večjo zanesljivost in prenosljivost kot uporaba standardnih orodij (*requirements.txt* in *django.config*).

## 2.4 Baza uporabnikov

Eno izmed največjih težav pri razvoju je predstavljal dostop do podatkov o dijakih. Za pravilno delovanje aplikacije je bil potreben dostop do imena in priimka, e-poštnega naslova in oddelka vsakega izmed dijakov na šoli.

Ker so v tem času na šoli načrtovali prehod na prijavni sistem *Arnes AAI*<sup>4</sup>, je bil prvotni načrt uporaba le-tega za prijavo dijakov v sistem. Zaradi težavnosti migracije podatkov vseh dijakov v nov sistem pa se je ta proces na šoli zavlekel do te mere, da ga za prvi preizkus nismo mogli uporabiti.

Da smo preizkus lahko vseeno izvedli, smo se odločili uporabiti kar šolske e-poštne naslove, ki so gostovani v Microsoftovem oblaku. Ta omogoča prijavo preko dobro standardiziranega protokola OAuth2 za vse račune v mreži Microsoft Graph.

<sup>4</sup> Enotni prijavni sistem za šole in univerze po celi Sloveniji – <https://aai.arnes.si/>

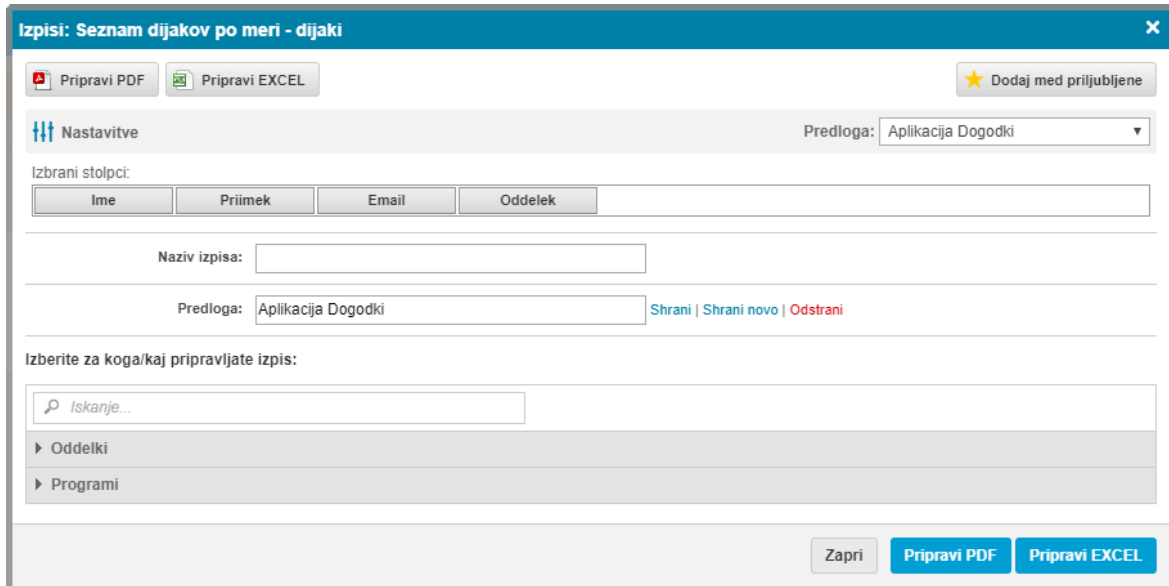
Za integracijo smo uporabili odprto-kodno knjižnico *python-social-auth* in pripadajoč Django modul. Po tem, ko smo na Microsoftovem portalu za razvijalce registrirali aplikacijo in ustvarili dostopne ključe, smo lahko relativno enostavno omogočili dijakom, da se prijavijo s svojimi šolskimi e-poštnimi naslovi.

Dostop do podatkov o dijakih (predvsem do oddelkov) pa je predstavljal večjo težavo. V idealnem primeru naša aplikacija ne bi shranjevala kakršnih koli podatkov o dijakih in bi do njih le dostopala iz šolske podatkovne baze. To bi pomenilo, da bi bili podatki vedno točni in najnovejši. Sočasno s preходом na Arnes AAI se je na šoli načrtovala tudi nova LDAP baza, v katero bi bili vpisani vsi uporabniki. S pomočjo Django modula *django-ldapdb* bi tako lahko relativno enostavno dostopali do podatkov o uporabnikih kot da bi bili shranjeni v lokalni bazi.

Ker pa se je tudi ta proces zavlekel, dostopa do baze dijakov do roka za oddajo nismo dobili, zato smo morali podatke najti drugje.

Vsi podatki o dijakih so shranjeni v sistemu eAsistent, ki žal ne omogoča programskega (API) dostopa, na voljo pa nam je funkcija za izvoz določenih podatkov v preglednico. Čeprav to ni najbolj zanesljiv ali enostaven postopek, smo morali vsaj začasno omogočiti ročni uvoz podatkov iz eAsistenta.

V vmesniku eAsistent je možno določiti polja, ki jih v končni datoteki želimo (v našem primeru ime, priimek, e-pošto in oddelek), ter seznam dijakov (po oddelkih), ki jih želimo zajeti (v našem primeru vse). K sreči je to konfiguracijo možno shraniti kot predlogo in s tem administratorjem delo vsaj malo olajšati.



Slika 5: Izvoz podatkov iz eAsistenta

Želene podatke dobimo v obliki Office Open XML preglednice (.xlsx). Prva vrstica preglednice je prazna, saj daje prostor logotipu eAsistent, šele v drugi vrstici pa so glave stolpcev, ki ji sledijo vrstice s podatki.

Pri uvozu podatkov smo se odločili za še en odprtokoden Django modul *django-import-export*. Ta v ozadju uporablja knjižnico *tablib*, ki ima vgrajeno podporo za takšne datoteke.

Skozi testiranje pa smo odkrili dva problema:

1. prazna vrstica z logotipom prepreči samodejno branje imen stolpcev (saj so ta tradicionalno napisana v prvem) in
2. v formatu `.xlsx` bralnik pogosto ne zazna konca tabele in nadaljuje z branjem praznih vrstic (predvidevamo, da zaradi nespoštovanja standarda z obeh strani).

Čeprav bi v teoriji lahko datoteke pred uvozom samodejno popravili, smo se na koncu odločili, da se tega ne bomo lotili, saj bi s tem programu dodali še več nepotrebnih sistemskih zahtev, rešitev z uvažanjem pa je tako ali tako le začasna.

Na koncu smo ugotovili, da je starejši Microsoftov format `.xls` veliko bolj zanesljiv, zato smo uvažanje omogočili le na tega, na uvozni zaslon pa napisali navodila za pripravo datoteke.

Uvoz

**Navodila za uporabo:**

Opozorilo:  
Zaradi nezanesljivosti bralnikov datoteke `xlsx` (Microsoft Excel) niso podprte. **Trenutno je podprt samo format `xls` (Microsoft Excel 2003).**

Priprava datoteke  
eAsistent podatke izvaža v nestandardnem formatu, zato jih je pred uvozom potrebno pripraviti:

1. Prenešeno datoteko iz eAsistenta odprite v programu za urejanje preglednic (npr. Microsoft Excel)
2. **Izbrišite eAsistent logotip**
3. Izbrišite celotno prvo (prazno) vrstico
4. Datoteko shranite v enem izmed podprtih formatov

Uvoz:  
Izberite datoteko za uvoz, v spustnem meniju izberite ustrezen format in kliknite gumb "Uvozi".

Ko se vam prikaže tabela preverite, če so podatki pravilni. Ko ste v to prepričani, uvoz potrdite z gumbom "Potrdi uvoz".

---

This importer will import the following fields: Ime, Priimek, Oddelek, Email, username

Datoteka preglednice:  No file selected.

Format:

Slika 6: Uvoz podatkov iz preglednice

Ko administrator datoteko naloži, ima možnost spremembe še pregledati, preden se shranijo v podatkovno bazo.

Ta postopek je potrebno opraviti z vsakim novim šolskim letom ter ob vsaki spremembi podatkov o dijakih (prepisi, vpisi, izpisi ...).

## 2.5 Gostovanje

Za gostovanje aplikacije so potrebni trije deli:

- statični spletni strežnik, ki servira vse statične datoteke (CSS, JS, slike) in datoteke, ki so jih naložili uporabniki,
- aplikacijski strežnik, ki odgovarja na vse ostale zahteve brskalnika in
- strežnik podatkovne baze.

Prvi preizkusi aplikacije so potekali na enem izmed lastnih strežnikov (virtualni strežnik s sistemom Ubuntu Server 16.04). Ker v tistem času aplikacija še ni bila dovolj stabilna, smo jo v začetku na strežniku gostovali kar neposredno iz izvorne kode najnovejše revizije Git. S pomočjo skripte, ki jo je Git repozitorij na strežniku zagnal, ko je prejel novo revizijo (`git-hooks/post-receive`), smo poskrbeli, da se je vsaka nova revizija uveljavila na disku (`git checkout`), morebitne migracije izvedle (`manage.py migrate`) in strežnik ponovno naložil (`systemctl reload apache2`).

Samo aplikacijo kot tudi serviranje statičnih datotek sta poganjala spletni strežnik *Apache* in vgrajen modul *mod\_wsgi*. Za ločitev Python okolja od systemskega smo ponovno uporabili orodje *venv*, vse skupaj pa je v Apache konfiguraciji izgledalo približno tako:

```
# Pot do vstopne točke WSGI
WSGIScriptAlias / /var/www/dogodki/ip-2019/dogodki_core/wsgi.py
# Ločitev procesa od ostalih in aktivacija virtualnega okolja (venv)
WSGIDaemonProcess ip19 python-home=/var/www/dogodki/venv/ python-
path=/var/www/dogodki/ip-2019
WSGIProcessGroup ip19
WSGIApplicationGroup %{GLOBAL}
# Serviranje statičnih datotek
Alias /static/ /var/www/dogodki/static/
Alias /media/ /var/www/dogodki/media/
```

Za gostovanje podatkovne baze smo uporabili strežnik *MariaDB*, saj je ta bil že postavljen, v prihodnosti pa bomo uporabili strežnik *PostgreSQL*, ki je priporočen s strani razvijalcev sistema Django in tako veliko bolje podprt.

Ker pa v prihodnosti želimo uporabo aplikacije omogočiti več šolam, smo se aplikacijo odločili zapakirati s pomočjo tehnologije *Docker*. Ta iz definicijske datoteke (`Dockerfile`) zgradi sliko celotnega uporabniškega dela operacijskega sistema (ang. *userspace*), ki jo zažene v lastnem kontejnerju. Kontejnerji so tehnologija jedra Linux, ki omogočajo izolacijo aplikacij do podobne mere kot systemska virtualizacija, le da ne emulirajo strojne opreme, na kateri bi delovalo jedro OS, ampak uporabijo kar jedro tekočega operacijskega sistema.

Ker pa za pravilno delovanje potrebujemo več kot eno aplikacijo, smo naš kontejner povezali s še enim, ki bo serviral statične datoteke (`nginx`) in enim za gostovanje podatkovne baze (`postgresql`). Vse skupaj smo definirali in postavili z orkestracijskim orodjem *docker-compose*





## 2.6 Varnost in zasebnost

Seveda smo posebno pozornost posvetili tudi varnosti aplikacije.

Začeli smo na nivoju povezave, kjer smo vso komunikacijo z uporabnikom zaščitili z SSL certifikatom. Za pridobitev brezplačnega certifikata smo uporabili storitev *LetsEncrypt*<sup>5</sup>, ki omogoča samodejno verifikacijo lastništva domene in samodejno podaljševanje certifikatov.

Sistem Django je bil narejen z varnostjo v mislih, zato so vsi deli naše aplikacije, ki jih popolnoma poganja Django (torej celotni administratorski vmesnik, upravljanje računov in procesiranje obrazcev), zadostno zavarovani. Django ima vgrajeno zaščito proti pogostim napadom, kot so *SQL injection*, *Cross-Site Scripting*, *Cross-Site Request Forgery* in drugi, piškotki seje so kriptografsko podpisani, interni sistem dovoljenj pa omejuje dostop do administratorskih orodij le določenim uporabnikom.

Tudi za varno hrambo gesel bi poskrbel Django, a ker naša aplikacija uporablja zunanjo prijavo in zato gesel uporabnikov nikoli ne vidi, v našem primeru to niti ni potrebno.

Edini podatki, ki jih aplikacija hrani, so imena, priimki in oddelki dijakov ter njihove prijave na dogodkih. Ker je aplikacija gostovana na šolski infrastrukturi in pod izključnim nadzorom šolskega systemskega administratorja, dodatno soglasje pod zakonodajo GDPR ni potrebno. Zaradi neobčutljive narave teh podatkov pa tudi enkripcije shrambe nismo implementirali.

V vsakem primeru pa je dostop do aplikacije omejen le na prijavljene uporabnike pod ustrezno domeno. Na zunanji strani je omogočen le dostop do vmesnika aplikacije preko kriptirane HTTP povezave, vse druge storitve (podatkovna baza, beleženje napak ipd.) pa so s požarnim zidom omejene na interno omrežje. Tretje osebe torej ne morejo videti kakršnih koli podatkov, dijaki pa lahko vidijo le prijave njihovega oddelka na posameznih dogodkih.

<sup>5</sup> <https://letsencrypt.org>

### 3 UPORABA APLIKACIJE

Aplikacija je razdeljena na dva dela – administratorski in uporabniški vmesnik. Priprava dogodka se začne na administratorskem, do katerega imajo dostop le profesorji, ki te dogodke organizirajo.

Po prijavi s šolskim računom dobijo dostop do obrazca za ustvarjanje dogodka. Tam izpolnijo podatke, kot so naslov dogodka, datum, rok za prijavo in kratek opis dogodka. V istem obrazcu lahko urejajo tudi skupine. Nove skupine lahko poljubno dodajajo in spreminjajo, določijo največje dovoljeno število prijav iz enega oddelka, v opis pa vpišejo vse pomembne informacije, ki jih dijaki potrebujejo (npr. lokacija, čas začetka, potreba oprema ...).

Dodaj Dogodek

Naslov:	<input type="text" value="Zimski športni dan"/>
Datum:	<input type="text" value="04.02.2019"/> Danes
Rok prijave:	Datum: <input type="text" value="31.01.2019"/> Danes Ura: <input type="text" value="18:00"/> Takoj
Opis:	<div style="border: 1px solid #ccc; padding: 5px;">Zimski športni dan za vse dijake, razen zaključnih letnikov. Prisotnost je obvezna. V primeru opravičljive odsotnosti se s koordinatorjem dogovorite za drugačno pridobivanje ur OIV.</div>
<b>SKUPINE</b>	
Skupina: #1	
Naslov:	<input type="text" value="Smučanje"/>
Opis:	<div style="border: 1px solid #ccc; padding: 5px;">Zbor pri spodnji vzpenjači pod Pohorjem ob 8:00. Cena karte: 8€ Izposoja smuč: 5€ Predviden konec: 12:00</div>
Število mest:	<input type="text" value="5"/>
Skupina: #2	
Naslov:	<input type="text" value="Drsanje"/>
Opis:	<div style="border: 1px solid #ccc; padding: 5px;">Zbor pri Ledni Dvorani ob 8:00 Cena karte: 3€ Izposoja drsalk: 3€ Predviden konec: 11:00</div>
Število mest:	<input type="text"/>
<a href="#">+ Dodaj še eno skupino</a>	
<input type="button" value="SHRANI"/>	

Slika 7: Vmesnik za urejanje dogodka

Ko je dogodek ustvarjen, lahko na vmesniku za povabilo dijakov izberejo oddelke, ki jim je dogodek namenjen. S tem sistem izbranim dijakom omogoči dostop do dogodka ter na šolski e-poštni naslov pošlje sporočilo s povezavo za prijavo.



Slika 8: E-poštno sporočilo, ki ga prejmejo dijaki

Dijaki se po kliku na povezavo morajo v aplikacijo prijaviti (za kar je po navadi potreben le en klik, saj so v šolski račun zaradi e-pošte že prijavljeni), nato pa dobijo na izbiro seznam skupin, ki so jim na voljo. S klikom na gumb »Prijava« se v skupino prijavijo in so s tem svoje delo opravili.

Na administratorski strani lahko profesorji spremljajo prijave dijakov. S filtri na desni strani lahko določijo, da jim sistem prikaže le dijake iz določenega oddelka, skupine ali pa tiste, ki skupine še niso izbrali.

<input type="checkbox"/>	DOGODEK	UPORABNIK	SKUPINA	ODDELEK
<input type="checkbox"/>	Zimski športni dan	Ahac Rafael Bela	-	2.ag
<input type="checkbox"/>	Zimski športni dan	Vito Zupanič	-	2.ag
<input type="checkbox"/>	Zimski športni dan	Mia Vehabović	-	2.ag
<input type="checkbox"/>	Zimski športni dan	Klemen Skok	-	2.ag
<input type="checkbox"/>	Zimski športni dan	Laren Sedlanič	-	2.ag
<input type="checkbox"/>	Zimski športni dan	Maksim Rozman	-	2.ag
<input type="checkbox"/>	Zimski športni dan	Gašper Rebernak	-	2.ag
<input type="checkbox"/>	Zimski športni dan	Luka Primec	-	2.ag
<input type="checkbox"/>	Zimski športni dan	Dominik Pignar	-	2.ag
<input type="checkbox"/>	Zimski športni dan	Nejc Novak	-	2.ag
<input type="checkbox"/>	Zimski športni dan	Nik Nešović	-	2.ag

**FILTER**

Po oddelku

- Vse
- 1.ag
- 2.ag
- 4.ag
- prof
- 

Po skupini

- Vse
- Brez
- Zimski športni dan: Smučanje
- Zimski športni dan: Drsanje
- Zimski športni dan: Bowling
- Zimski športni dan: Plavanje
- Zimski športni dan: Nogomet
- Zimski športni dan: Namizni tenis
- Zimski športni dan: Šah
- Zimski športni dan: Pohod
- Zimski športni dan: Fitnes

Slika 9: Pregled prijavljenih dijakov

S klikom na posamezen vnos pa lahko administrator tudi spremeni podatke prijave.

## 4 ANALIZA IZDELKA

Da bi ugotovili uspešnost projekta, smo se vrnili k ciljem, ki smo jih določili v začetku.

Glede uporabniškega vmesnika smo svoj cilj dosegli. Vsi dijaki, ki so bili del testne skupine, so se na športni dan prijavili brez težav ter brez kakršnih koli navodil – razen »poglejte na šolski e-poštni naslov«. Tudi profesorji, ki bodo poleg glavnega uporabljali še administratorski vmesnik, so se na aplikacijo zelo hitro navadili.

Glede integracije svojega pravega cilja nismo popolnoma dosegli. Sistem sicer je integriran s šolsko infrastrukturo do mere, ki jo vidijo dijaki, v ozadju pa integracija še ni takšna, kot bi jo želeli. Razlogi za to niso pod našim nadzorom in so predvsem posledica precej omejenega časa. Uvažanje podatkov s pomočjo preglednic je sicer dobra začasna rešitev, a še vedno načrtujemo polno integracijo z imenikom LDAP, ko bo ta na voljo. Hkrati pa bo možnost ročnega uvažanja vseeno koristila v prihodnosti, če aplikacijo želimo uporabiti na šoli, ki centralnega imenika sploh nima.

Uporaba ogrodja Bootstrap je poskrbela, da aplikacija izgleda konsistentno na širokem razponu naprav; z uporabo tradicionalnih mehanizmov za pošiljanje podatkov (standardni HTML obrazci popolnoma brez potrebe za JavaScript) pa smo zagotovili, da bo aplikacija pravilno delovala tudi v starejših brskalnikih in na počasnejših napravah. Na več različnih napravah (pametni telefoni, tablica, računalnik) in brskalnikih (Firefox 50 in 65, IE 11 in 10, Chromium 72, Edge 44) je aplikacija delovala brezhibno, zato menimo, da smo tretji cilj dosegli.

Tudi primerjava z dosedanjimi preglednicami na papirju je pokazala, da smo izpolnili vse zahteve. Dijaki imajo na voljo vse podatke, ki so jih imeli na papirju, prijave pa lahko brez težav kasneje tudi spreminjajo. Profesorji imajo preko administratorskega vmesnika pregled nad dijaki celo boljši kot prej, upravljajo pa lahko tudi vse prijave dijakov.

Glede same stabilnosti aplikacije pa odločitve še ne moremo narediti. Zaradi časovne omejitve smo izvedli le en večji praktični preizkus, ki je sicer stekel brez večjih težav, določenih situacij pa nismo uspeli preizkusiti. V testnem okolju smo sicer simulirali menjavo šolskega leta, spremembo oddelka in nekaj možnih načinov goljufanja pri prijavi, a le uporaba v daljšem časovno obdobju bo pokazala, do kakšne mere je aplikacija zanesljiva.

## 5 DRUŽBENA ODGOVORNOST

Digitalizacija je smer, v katero se premikajo praktično vsi deli modernega življenja in šolstvo pri tem ni nobena izjema. Sistemi, kot je eAsistent, olajšajo vodenje ocen in prisotnosti, spletne učilnice pa vodenje pouka in učnega gradiva. Šole pa zraven rednega pouka izvajajo še mnoge druge dejavnosti, kot so krožki, športni dnevi in obvezne izbirne vsebine. To so področja, ki se v večini šol še niso digitalizirala.

Ročno vodenje prijav na dogodke je za profesorje zelo zamudno. Od pouka si morajo vzeti čas, da zbirajo prijave dijakov, ki se nato še dolgo časa odločajo. Prijave morajo ročno šteti, iskati manjkajoče in voditi morebitne naknadne spremembe. Z uporabo digitalnega sistema s prijavi po razpisu dogodka nimajo skoraj nič dela.

Ker pa dogodke po navadi organizira več kot en profesor, pogosto prihaja tudi do problemov z deljenjem podatkov. Fotokopije seznamov s prijavi so hitro zastarele, sortiranje podatkov iz ločenih tabel za posamezne oddelke v tabele, urejene po skupinah, pa je prav tako zamudno. Z našim sistemom imajo vsi organizatorji možnost pregleda ažurnega stanja prijavi in to v formatu, ki jim najbolj ustreza. Profesorji v posameznem oddelku lahko preverijo prijave za njihove dijake, vodje posameznih skupin pa lahko na enem mestu vidijo vse dijake šole v njihovi skupini.

Nazadnje pa naš sistem naredi celoten proces prijave tudi bolj pravičen. Nedovoljeno spreminjanje prijavi med dijaki je sicer redko, a se dogaja. Varnostni mehanizmi v naši aplikaciji to preprečujejo. Ker je spletni vmesnik dijakov dostopen kadarkoli in kjerkoli, imajo dijaki tudi dovolj časa, da se dogovorijo med seboj, hkrati pa omogoča prijavo tudi dijakom, ki jih zaradi bolezni ali drugih razlogov tisti dan ni bilo pri pouku.

## 6 ZAKLJUČEK

Ob začetku projekta smo si zadali cilj, da na naši šoli rešimo konkreten problem, ki je tratil čas profesorjev in dijakov. Ta cilj smo dosegli in že v kratkem bo naša aplikacija popolnoma nadomestila zamudno vodenje prijav na papirju.

Po uspehu na naši šoli pa želimo razvoj osredotočiti na pripravo aplikacije za uporabo tudi na drugih šolah – predstavniki nekaterih šol so namreč že izrazili željo za preizkus aplikacije. Zaradi zakonodaje o varstvu podatkov, bi radi šolam omogočili gostovanje aplikacije na lastni infrastrukturi, zato je olajšanje namestitve in integracije naš prvi cilj za nadaljnji razvoj. Najpomembnejši del tega sta integracija prijave s sistemom Arnes AAI in baze uporabnikov z imeniki LDAP, s čimer smo že začeli.

Glede prihodnosti aplikacije še nimamo konkretnega načrta. Po pogovoru z bližnjimi šolami želimo ugotoviti najučinkovitejši razvojni model aplikacije – trenutno se nagibamo proti odprto-kodni licenci, ki bi vsem šolam omogočala skupno nadaljevanje razvoja, če pa šole za to nimajo interesa pa razmišljamo tudi o možnosti postavitve storitve v oblaku.

Več informacij o nadaljnjem razvoju aplikacije, vključno z izvorno kodo in delujočo demonstracijo, lahko najdete na spletni strani projekta:



<https://rebrand.ly/ip2019-dogodki>

## 7 ZAHVALE

Na prvem mestu se zahvaljujem mentorjem za pomoč pri načrtovanju in preizkušanju aplikacije. Projekt prav tako ne bi uspel brez pomoči šolskega systemskega administratorja, ki nam je priskrbel prostor na šolskem strežniku za gostovanje aplikacije in vseh v vodstvu šole, ki so projekt podprli. Zahvalo pa si zaslužijo tudi dijaki vseh oddelkov, ki so sodelovali v prvih preizkusih aplikacije.

### 7.1 Uporaba odprte kode

Celoten projekt se na mnogih mestih zanaša na odprto-kodno programsko opremo. Zahvaljujem se vsem, vključenim v naslednje odprto-kodne projekte:

- **GNU in Linux** – operacijski sistem GNU/Linux, na katerem je bila aplikacija razvita (**Arch Linux**) in ki poganja strežnike, na katerih je gostovana (**Alpine Linux**),
- **Git** – sistem za vodenje sprememb in upravljanje z različicami,
- **GitLab** – spletna aplikacija za gostovanje repozitorijev Git in Docker, avtomatizirano izgradnjo in vodenje projekta,
- **Visual Studio Code** – urejevalnik kode/integrirano razvojno okolje
- **Python** – programski jezik, ki poganja backend aplikacije,
- **Pipenv** – orodje za upravljanje Python projektov,
- **Django** – ogrodje backenda aplikacije,
- Django moduli **django-environ**, **django-social-auth** in **django-import-export**,
- **Bootstrap** – ogrodje frontenda aplikacije,
- **FontAwesome** – ikone na frontendu aplikacije,
- **SQLite** – lokalna podatkovna baza v času razvoja,
- **MariaDB** – podatkovna baza na testnem strežniku,
- **PostgreSQL** – podatkovna baza na končnem strežniku,
- **Docker/Moby** –
- **Sentry** – sistem za obveščanje o napakah,
- **Apache HTTP server** – HTTP in aplikacijski strežnik,
- **Nginx** – strežnik statičnih datotek,
- **Gunicorn** – aplikacijski strežnik.

## 8 VIRI IN LITERATURA

Open Web Application Security Project. (2017). *Security Top 10*. Pridobljeno iz OWASP: [https://www.owasp.org/index.php/Top\\_10-2017\\_Top\\_10](https://www.owasp.org/index.php/Top_10-2017_Top_10)

Udeleženci projekta Django. (2018). *Django documentation*. Pridobljeno iz Django project: <https://docs.djangoproject.com/en/2.1/>

Udeleženci projekta Django. (2018). *Security in Django*. Pridobljeno iz Django documentation: <https://docs.djangoproject.com/en/2.1/topics/security/>

Wiggins, A. (2017). Pridobljeno iz The Twelve-Factor App: <https://12factor.net/>