PD.TVZ.HR
POLYTECHNIC & DESIGN Vol. 10, No. 2, 2022.
Mrežna inačica: ISSN 2459-6302
Tiskana inačica: ISSN 1849-1995
DOI: 10.19279/TVZ.PD.2022-10-2-05

RAZVOJ APLIKACIJE ZA UČENJE BRZOG ČITANJA

DEVELOPMENT OF A SPEED READING LEARNING APPLICATION

Alen Šimec[1], Mate Mišić[1], Davor Lozić[2]

Tehničko veleučilište u Zagrebu, Vrbik 8, 10 000 Zagreb, Hrvatska
Veleučilište Velika Gorica, Vrbik 8, Zagrebačka ulica 5, 10410 Velika Gorica, Hrvatska

SAŽETAK

Sprint-material je internet aplikacija koja je kreirana kako bi omogućila vježbe brzog čitanja. Trenutno postoji nekoliko sličnih aplikacija koje omogućavaju funkcionalnost brzog čitanja teksta, ali te aplikacije ne podržavaju trajno spremanje sadržaja što zahtjeva često otvaranje dokumenata na lokalnom računalu kako bi se kopirao tekst na stranicu. Drugi značajni problem je što ne podržavaju font za disleksiju ili mogućnost korištenja različitih fontova. Takvo ograničenje gotovo da onemogućuje korištenje funkcionalnosti stranice osobama koje boluju od disleksije. Posljednji problem je to što postojeće stranice ne omogućavaju dijeljenje sadržaja, pa je potrebno organizirati distribucijski kanal kako bi se štivo dostavilo svim zainteresiranim osobama.

Ključne riječi: *Sprint-material, aplikacija za učenje brzog čitanja, čitanje, brzo čitanje, razvoj*

ABSTRACT

Sprint-material is an internet application that was created to enable speed reading exercises. There are currently several similar applications that provide speed-reading functionality, but these applications do not support permanent content storage, which requires frequently opening documents on a local computer to copy text to a page. Another significant problem is that they do not support a dyslexia font or the ability to use different fonts. Such a restriction makes it almost impossible for people with dyslexia to use the site's functionality. The last problem is that the existing pages do not allow content sharing, so it is necessary to organize a distribution channel in order to deliver the reading to all interested persons.

KeyWords: *Sprint-material, application for learning speed reading, reading, speed reading, development*

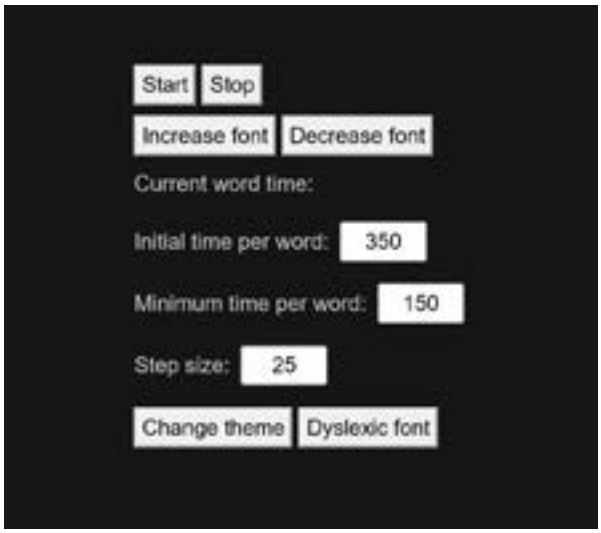
1. UVOD
1. INTRODUCTION

Aplikacija je bazirana na Java programskom jeziku i Spring programskom okviru, a dizajnirana je MVC (Model-View-Controler) arhitekturom. Sastoji se od modela, repozitorija, servisa i kontrolera. Modeli predstavljaju objekte (entitete) koji sadrže podatke potrebne za sve funkcionalnosti aplikacije. Repozitoriji služe za komunikaciju sa bazom podataka, dohvaćanje podataka, zapisivanje, izmjene na postojećim podatcima te brisanje, a servisi obrađuju dohvaćene podatke i pripremaju podatke za zapisivanje u bazu te koriste funkcionalnosti repozitorija. Kontroleri kreiraju predloške popunjene podatcima koje su pripremili servisi. Neke od bitnih funkcionalnosti aplikacije su:

- korisničke uloge
- kreiranje i izmjena grupa
- komunikacija unutar grupe
- objava tekstova za vježbu brzog čitanja
- font prilagođen osobama sa disleksijom

Aplikacija razlikuje dvije korisničke uloge korisnik i moderator , a razlika je u što editor može kreirati javne grupe koje su vidljive i u koje se mogu, bez posebnih zahtjeva, priključiti svi korisnici. Svi korisnici sa ulogom korisnika mogu kreirati grupe u koje će pozvati bilo kojeg drugog korisnika aplikacije koristeći se njihovim nadimkom, a također može i maknuti korisnika

kojeg više ne želi u grupi. Kako bi se korisnik mogao priključiti grupi mora prihvatit poziv kreatora grupe. Kreator grupe u grupu može dodati tekstove za vježbu brzog čitanja koji su vidljivi svim korisnicima unutar grupe. Svaki korisnik u grupi može napisati objavu unutar grupe koja će biti vidljiva svim korisnicima unutar navedene grupe. Tekstovi za brzo čitanje nalaze se unutar grupa, te ih objavljuje kreator grupe. Tekstovi se pokreću na posebno prilagođenom ekranu koji omogućuje dvije teme tamnu i svijetlu. Pri pokretanju teksta određuje se početna i krajnja brzina čitanja te ubrzanje do krajnje brzine. Također pri pokretanju tekstova može se izabrati font prilagođen osobama koje imaju problema sa prepoznavanjem slova.



Slika 1 *Forma za unos parametara strojnog čitanja*

Figure 1 *Form - parameter input for machine reading*

2. TEHNIKA ZA RAZUMIJEVANJE I PROCESIRANJE PISANOG JEZIKA

2. TECHNIQUE FOR UNDERSTANDING AND PROCESSING WRITTEN LANGUAGE

Rapid Serial Visual Presentation RSVP je započelo kao istraživanje pozornosti kasnih 1950-ih, a Forster je 1970 primijenio tehniku na razumijevanje i procesiranje pisanog jezika. RSVP se sastoji od prikazivanja jedne ili više riječi na istom mjestu izmjenjujući ih periodički. Tako se smanjuje kretanje oka i povećava koncentracija.[3] Istraživanja su pokazala da su određene karakteristike optimalne, a to su da je širina teksta u prosjeku 12 znakova, do tri riječi po iteraciji, korištenje teksta umjesto natuknica i prikaz prazne iteracije između rečenica. Tijekom istraživanja klasično čitanje s papira je bilo brzinom 156 do 402 riječi po minuti s medijanom od 290 dok je kod korištenja RSVP metode bilo brzinom 178-529 riječi po minuti s medijanom od 348. (Applying the Rapid Serial Visual Presentation Technique to Small Screens (2000) by Karin Sicheritz , Language Engineering Programme, Lars Borin) RSVP se i dalje istražuje u različite svrhe tako da je postalo popularno koristiti drugačiju boju za srednje slovo kako bi se oči lakše centrirale na riječ.

3. KORIŠTENJE TEHNOLOGIJA OTVORENOG KODA

3. USING OPEN SOURCE TECHNOLOGIES

U izradi projekta korištene su isključivo tehnologije otvorenog koda (engl. *open source*). Projekt se na najvišoj razini sastoji od aplikacije i baze podataka. Programski jezik Java je odabran za izradu aplikacije zato što je podržan na različitim operacijskim sustavima, a Spring Boot framework je industrijski standard pa nije bilo razloga za odabir nečega drugoga.

Pristup bazi je generaliziran unutar aplikacije što omogućava korištenje bilo koje podržane relacijske baze podataka poput MariaDB, MySQL, PostgreSQL, H2 ili IBM DB2. U razvoju i testiranju je korištena baza H2 zbog podrške rada „u memoriji“ (engl. *in-memory*), pa je nije potrebno instalirati na računalo na kojem se programira. Podatci nisu zadržani između pokretanja što je čini dobrom za ovaj projekt. U kasnijim fazama je baza zamijenjena s MariaDB bazom instaliranom na poslužitelju (engl. *server*) kako bi aplikacija zadržala podatke između pokretanja.

Java aplikacija koristi Spring programski okvir kao osnovnu tehnologiju što omogućava jednostavnu raspodjelu na slojeve. Za pristup bazi korišten je Spring Data, a za prikaz Thymeleaf koji su detaljnije opisani u nastavku.

Javascript je korišten u kombinaciji s HTML-om (engl. *Hypertext Markup Language*) i CSS-

om (engl. *Cascading Style Sheets*) kako bi se podaci mogli prikazati korisniku unutar Internet preglednika.

4. PROGRAMSKI OKVIR PROGRAMSKOG JEZIKA JAVA

4. JAVA PROGRAMMING LANGUAGE FRAMEWORK

Spring je programski okvir programskog jezika Java, koji se u pravilu koristi za razvoj internetskih aplikacija. Spring radi na programerskom principu IoC (engl. *inversion of control*) kontejnera koji omogućava konfiguriranje i upravljanje objektima. IoC kontejneri su odgovorni i za upravljanje životnim ciklusima kreiranih objekata, te pozivanje metoda inicijalizacije. IoC je baziran na tehnici "dependency injection" u kojoj objekt prima druge objekte o kojima je ovisan, a ovisnosti se definiraju uz pomoć anotacija.

Springboot je open-source programski okvir koji omogućava izradu samostalnih aplikacija baziranih na Javi programskom jeziku i Spring programskom okviru. Olakšava sam razvoj i produkciju aplikacija zbog minimalnih konfiguracija i olakšanim upravljanjem vanjskih ovisnosti aplikacije.

Spring Data je dio Spring programskog okvira čija je namjena konzistentan pristup podatkovnom sloju bez obzira na implementacijske detalje podatkovnog sloja uz moguće korištenje prednosti različitih tehnologija. Sastoji se od modula koji se mogu uključiti ovisno o potrebama projekata, a neki od njih su: JDBC, JPA, LDAP, Redis, Apache Cassandra, Apache Hadoop, Elasticsearch.[5] Spring Data JPA je modul za olakšanu implementaciju metoda za pristup podatkovnom sloju relacijske baze koristeći sučelja. Nasljeđivanjem jednog od JPA sučelja dobije se pristup CRUD operacijama i paginaciji za određeni entitet koristeći parametriziranu vrstu (eng. *generics*).[6] Kod pisanja sučelja mogu se koristiti ključne riječi za opisivanje upita. Ključne riječi su: Distinct, And, Or, Is, Equals, Between, LessThan, LessThanEqual, GreaterThan, GreaterThanEqual, After, Before, IsNull, Null, IsNotNull, NotNull, Like, NotLike, StartingWith, EndingWith, Containing, OrderBy, Not, In, NotIn, True, False, IgnoreCase.[11]

U slučaju da ključne riječi nisu dovoljne za opis željenog upita, moguće je pisati vlastiti SQL. Prvi način za pisanje vlastitog SQL koda je korištenjem anotacije @NamedQuery na entitetskoj klasi.

Ovakav pristup se koristi kod jednostavnih programa i nije preporučen za kompleksne upite i velike programe zato što miješa entitet i pristup podatkovnom sloju. Preporučeni pristup je korištenje @query anotacije u repozitorijskom sučelju. Upiti se mogu pisati u JPQL ili SQL jeziku.

Sortiranje i paginacija se implementiraju dodavanjem PageRequest ili Sort parametra metodi. Sortirati se može po bilo kojem atributu entiteta ili po rezultatu funkcije nad entitetom ili njegovim atributom, ali to zahtjeva korištenje JpaSort.unsafe metode.

Unos korisničkih parametara je osiguran od napada SQL injection bez obzira koji od načina definiranja upita se koristi.[8]

Spring security je radni okvir za kontrolu pristupa u aplikacijama baziranim na Spring radnom okviru. Može služiti za autentifikaciju i autorizaciju korisnika ovisno o potrebama aplikacije. Samim uključivanjem ovisnosti (engl. *dependencies*), Spring Boot automatski zahtjeva prijavu korisnika za bilo kakvu interakciju s aplikacijom, generira prozor za prijavu, kreira korisnika s korisničkim imenom "user" i ispiše nasumičnu lozinku u konzolu za korištenje aplikacije, enkriptira lozinku BCrypt algoritmom, omogućuje odjavu korisnika, zahtjeva CSRF token kod POST zahtjeva, zaštićuje od tzv. session fixation napada i dodaje sigurnosna zaglavlja u svakom zahtjevu.[4]

Konfiguracija Spring security u implementaciji projekta konfigurira pristup različitim stranicama ovisno o prijavi, isključuje csrf na definiranim URL-ovima, definira URL za prijavu, odjavu i stranicu koja prikazuje odbijen pristup, te je promijenjena implementacija BCrypt algoritma iz $2A u $2Y što služi za potvrdu da je korištena implementacija s ispravljenom greškom u inicijalnoj implementaciji.

Pozivanje REST servisa je moguće na nekoliko načina, ali preporučeni način je korištenjem Web Client aplikacijskog sučelja. Prednost mu je podrška reaktivnom programiranju, te ne blokira HTTP zahtjeve (zahtjevi obrađeni asinhrono). Instanca se kreira koristeći WebClient.Builder kojeg Spring Boot automatski konfigurira za korištenje.

U implementaciji smo koristili metodu block() koja ne koristi reaktivno programiranje već blokira izvršavanje daljnjih naredbi dok se ne dohvati rezultat zato što je takav način bolje odgovarao.

Spring profili omogućavaju različitu konfiguraciju ovisno o profilu ili profilima s kojim pokrenemo aplikaciju. Može se konfigurirati od korištenja različitih klasa do promjene osnovnih parametara. Uključivanje servisa se konfigurira postavljanjem anotacije @Profile("imeProfila") iznad definicije klase, a isključivanje servisa se vrši na sličan način koristeći @Profile("!imeProfila").[11] Konfiguracija parametara se radi koristeći različite datoteke. Osnovna i uvijek uključena konfiguracija je u datoteci application.properties, dok se može za svaki profil dodatno definirati application-profile.properties. U implementaciji smo koristili dva profila, dev i prod tako da su postojale odgovarajuće datoteke application-dev.properties i application-prod.properties. U dev profilu je baza postavljena na H2 za lakši razvoj, a na prod profilu su definirane postavke za spajanje MariaDB bazu. U implementaciji su profili definirani kroz datoteke koje se kod razvoja aplikacije spremaju zajedno sa ostatkom programa. Takav pristup je korišten za jednostavnije pokretanje i mogućnost prijenosa parametara koristeći aplikaciju „git“.

Druga opcija je postaviti konfiguracijsku datoteku izvan aplikacije. Redoslijed pretrage konfiguracijske datoteke je "/config" direktorij u direktoriju gdje je aplikacija pokrenuta, te direktorij u kojem je aplikacija pokrenuta i na kraju "/config" direktorij u tzv. classpath. Ako niti jedna od tih lokacija ili format imenovanja datoteke ne odgovara potrebama projekta, moguće je postaviti parametar "--spring.config.location=" i u njemu napisati putanju do konfiguracijske datoteke bilo gdje na računalu. Koristeći drugi način moguće je definirati novi profil bez potrebe za novim pokretanjem i razvojem aplikacije.[6]

Spring Boot aplikacija se može pokretati na više načina, ali osnovni je tretirati ju kao običnu Java aplikaciju i koristiti Tomcat poslužitelj. U tom slučaju se pokreće na način "java -jar projectName.jar". U ovom slučaju kada profil nije postavljen koristi se profil "default" i konfiguracija iz datoteke application.properties. Za postavljanje profila potrebno je pokrenuti aplikaciju na način "java -jar projectName.jar --spring.config.name=profileName" ili "java -jar projectName.jar --spring.config.location=classpath:/biloKojeImeKonfiguracije.properties". U ovim primjerima je moguće postaviti n profila tako što ih odvojimo zarezom.[9]

Drugi način pokretanja aplikacije je korištenjem maven omotača tako što pokrenemo "mvn spring-boot:run", a profili se postavljaju koristeći "mvn spring-boot:run -Dspring-boot.run.profiles=profileName".[10]

Treći način pokretanja je zapakirati aplikaciju kao .war umjesto .jar i koristiti Apache Tomcat, Jetty, Undertow ili IBM WebSphere Liberty poslužitelj postavljen na razini računala. Takva konfiguracija se koristi u produkcijskom okruženju kod projekata zato što je tamo potrebna velika granuliranost prava i sigurnost aplikacije. Tada konfiguracija aplikacije više nije odgovornost programera već system administratora.[7]

Dohvat iznimki bez promjene konfiguracije, kada se dogodi situacija u programu koja nije dohvaćena, korisniku se prikazuje tzv. Whitelabel greška stranice (engl. *error page*) što nije dobar način za korisničko iskustvo (engl. *user experience*) i može biti sigurnosni problem ako se otkriju detalji implementacije.

U tu svrhu se definira klasa s anotacijom @ControllerAdvice koja sadrži logiku za prikaz različitih stranica ovisno o iznimci. U implementaciji smo odredili različiti prikaz za korisnike koji nisu autorizirani, te pogreške baze izazvane neispravnim zahtjevom i grupirali smo sve ostale iznimke zajedno.

Konfiguracija se vrši anotacijom @ExceptionHandler koja kao parametar prima klasu iznimke na koju se metoda odnosi. Kod iznimke

koja nije prije uhvaćena, te se provjeravaju sve metode dok se ne naiđe na najtočniju koja odgovara iznimci što znači da možemo koristiti svojstvo nasljeđivanja u objektno orijentiranoj paradigmi za definiranje pravila.

U implementaciji je kreirana iznimka UnauthorizedException koja nasljeđuje osnovnu RuntimeException i baca se kada korisnik nije autoriziran. Kod definiranja @ ExceptionHandler-a odvojene su metode za te iznimke tako da se poziva željena metoda. Ovakav pristup nam omogućava da metoda koja lovi RuntimeException služi kao univerzalno pravilo za sve ostale iznimke za koje nije definirano kako da se obrađuju.

Thymeleaf je jezik za predloške (engl. *templating engine*) programskog jezika Java. Temelji se na poslužiteljskom generiranju prikaznih predložaka za internetska i ne internetska okruženja. Za kreiranje predložaka koristi se HTML. HTML predloščima se podaci dodaju uz pomoć internacionalizacijskih i atributnih izraza. Thymleaf podržava i pozivanje metoda iz Utility klasa koje se koriste za manipulaciju tekstualnih, numeričkih i datumskih podataka. Kreirani predlošci predstavljanju pokazni (view) sloj u internetskim aplikacijama baziranih na MVC arhitekturi. HTTP zahtjev koji dolazi na kontroler pokreće dohvat podataka kojima se zatim popunjavanju unaprijed pripremljeni Thymeleaf predlošci. Izlaz iz kontrolera je popunjena HTML stranica koja je prikladna za prikaz u Internet preglednicima.[1]

Izgled sučelja nam omogućava određivanje često korištenih dijelova predložaka kao što su zaglavlje, navigacija i podnožje stranica, a po potrebi i druge dijelove vizualnog dijela stranice. Dijelovi predloška definiraju se i ugrađuju u predloške korištenjem predloška (engl. *Layout*) izraza, a definiraju se također uz pomoć HTML-a i Thymeleaf izraza, te podržavaju sve mogućnosti kao i čisti Thymeleaf predlošci.

i18n je skraćenica za pojam internacionalizacije u svijetu razvoja aplikacija. Internacionalizacija predstavlja mogućnost prevođenja svih tekstualnih podataka koji se nalaze na svim slojevima aplikacije. U programskom okviru Spring internacionalizacija provodi definiranjem podataka u tekstualnoj datoteci sa ekstenzijom .properties. Za svaki podržani jezik definira se po jedna takva datoteka. U datoteci se postavlja "ime" vrijednosti i vrijednost odnosno tekst koji će zamijeniti to ime na željenom mjestu.

5. IDEJE ZA UNAPRJEĐENJE PROJEKTA
5. IDEAS FOR PROJECT IMPROVEMENT

Vizualno aplikacija ne izgleda prema trenutnim općeprihavaćenim naputcima dizajnerske struke. Dizajn je odvojen od logike što omogućava zamjenu novim ekranima bez potrebe za promjenom logike.

Drugo poboljšanje je implementacija autentifikacije korištenjem vanjskog sustava za identifikaciju i autorizaciju korisnika koji podržava OAuth 2.0 standard. Time bi se izbjegla potreba za kreiranjem novih korisničkih računa ako već postoje u takvom sustavu, te mogučnost bržeg i jednostavnijeg korištenja korisničkih profila uz dodatnu sigurnosti prijave.

6. ZAKLJUČAK
6. CONCLUSION

Aplikacija je uspješno implementirana i funkcionalna. Javascript je korišten za logiku unutar korisničkog pretraživača. Postavke koje se mogu koristiti za promjenu brzine izmjene riječi i odabir veličine slova omogućava ugodno korištenje bez obzira na vještinu čitanja pojedinca. Promjena fonta na OpenDyslexic3 omogućava korištenje aplikacije za osobe s disleksijom ili srodnim bolestima.

Spring boot i ostale korištene tehnologije su dobro ispunile svoju zadaću i nisu predstavljale prepreku tokom razvoja. Odabrana arhitektura se može smatrati pretjerano kompleksnom zato što su podatkovni sloj, poslovna logika i korisničko sučelje odvojeni i nezavisni. Takav pristup je generalno poželjan, ali na aplikacijama manjeg obujma, te prednosti ne dolaze do izražaja, a potrebno je duže vrijeme programiranja.

Performanse sustava na velikoj količini korisnika

nisu izmjerene, ali aplikacija je vertikalno i horizontalno skalabilna..

7. REFERENCE
7. REFERENCES

[1.] Good, M.; Thymeleaf with Spring Boot; Kindle; ASIN:ƒB0795BQT6G; 2018

[2.] Nascimento, A. E.; OAuth 2.0 Cookbook: Protect your web applications using Spring Security; Packt Publishing; ISBN: 978-1788295963; 2017

[3.] Simone Benedetto, A. C.; Rapid serial visual presentation in reading: The case of Spritz https://www.researchgate.net/publication/270650767_Rapid_serial_visual_presentation_in_reading_The_case_of_Spritz; dohvaćeno 10.02.2021.

[4.] Splica, L.; Spring Security in Action; Manning Publications Co.; ISBN 9781617297731; 2020

[5.] Spring team; Spring; url: Reference documentation; https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#reference; Dohvaćeno: 12.03.2020.

[6.] Spring team; Spring; url: https://spring.io/projects/spring-data; Dohvaćeno: 12.03.2020.

[7.] Spring team; Spring; url: https://docs.spring.io/spring-boot/docs/1.0.1.RELEASE/reference/html/boot-features-external-config.html; Dohvaćeno: 14.03.2020. Dohvaćeno: 14.03.2020.

[8.] Spring team; Spring; url: https://docs.spring.io/spring-boot/docs/1.2.0.M1/reference/html/boot-features-profiles.html; Dohvaćeno: 18.03.2020

[9.] Spring team; Spring; url: https://spring.io/guides/gs/spring-boot; Dohvaćeno: 22.03.2020

[10.] Spring team; Spring; url: https://spring.io/projects/spring-boot#overview; Dohvaćeno: 28.03.2020

[11.] Walls, C.; Spring in Action; Manning Publications Co.; ISBN 9781617294945; 2019

AUTORI · *AUTHORS*

● **Alen Šimec** - rođen je 19.01.1979. U Zagrebu završava III. Gimnaziju, stručni dodiplomski studij na Tehničkom veleučilištu u Zagrebu (2002.), te stječe stručno zvanje inženjer informatike. Diplomirao je na stručnom dodiplomskom studiju na Visokoj školi za sigurnost, te je stekao zvanje diplomiranog inženjera sigurnosti (2006.). Diplomirao je na Specijalističkom diplomskom stručnom studiju politehnike, smjer informatika (2008). Doktorirao je na Filozofskom fakultetu Sveučilišta u Zagrebu, smjer Informacijske i komunikacijske znanosti, te stječe titulu Doktora znanosti, područje društvene znanosti, polje informacijske i komunikacijske znanosti.
Dr. sc. Alen Šimec sudjelovao je na predavanjima i razgovorima o unaprjeđenju suradnje tvrtke Microsoft s akademskom zajednicom, pohađao je i sudjelovao na stručnim i znanstvenim konferencijama sa radovima koji su objavljeni u tiskanom i digitalnom izdanju. Napisao je četiri knjige objavljene u Nacionalnoj sveučilišnoj biblioteci, te radi na Zagrebačkoj školi ekonomije i menagementa kao viši predavač. Stalni je zaposlenik na Tehničkom veleučilištu u Zagrebu, vlasnik je informatičke firme eBurza Grupa d.o.o., te je član neprofitne organizacije Rotary klub Zagreb.
Korespondencija · *Correspondence*
alen@tvz.hr

● **Davor Lozić -** predavač; programer. Predaje kolegije usko vezane uz informacijske sustave (baze podataka, operacijski sustavi - Linux) te obradom velikih količina podataka (baze u int. okruženju, uvod u umjetnu inteligenciju). Radio na izradi distribuiranih i visoko-dostupnih sustava, planiranju arhitekture i implementaciji CRM i CMS sustava. Recenzirao više radova i knjiga na konferencijama kao HEIC ili izdavače kao PacktPub.
Korespondencija · *Correspondence*
davor.lozic@vvg.hr