Obaveze odabranog ponuditelja u planiranju i razvoju IT rješenja

Sadržaj

Hrvatska akademska i istraživačka mreža – CARNET
Josipa Marohnića 5, HR - 10000 Zagreb · tel: +385 1 6661 616 · fax: +385 1 6661 615 · mail: ured@carnet.hr · www.CARNET.hr

1. Uvod

U dokumentu se nalaze obaveze ponuditelja za razumijevanje minimalnih tehničkih zahtjeva i pretpostavki Hrvatske akademske i istraživačke mreže (dalje u tekstu „CARNET“ ili „naručitelj“) koje se očekuju u planiranju i razvoju IT rješenja (dalje u tekstu „Rješenje“).

Dokumentom se opisuju osnovni koncepti IT arhitekture Rješenja, definiraju se ključni aspekti tehnološke arhitekture Rješenja, primjerice mikroservisna arhitektura, API management te način razvoja IT rješenja.

Dokument je prilog dokumentaciji o nabavi (dalje u tekstu „DoN“) usluga razvoja softvera te predstavlja opis tehničkih pretpostavki i arhitekture IT rješenja koje je funkcionalno opisano i naznačeno u DoN-u. U DoN-u mogu biti navedene i dodatne informacije i detalji vezani uz IT rješenje.

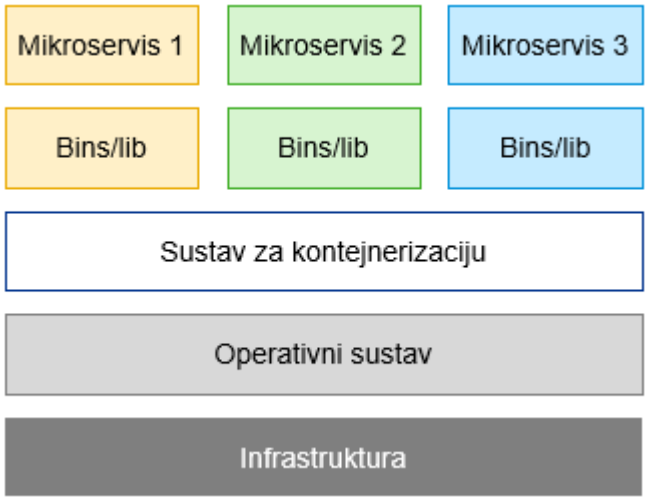
2. Mikroservisna arhitektura

Rješenje mora biti moguće mijenjati, nadograđivati, održavati i skalirati prema potrebama poslovanja s ciljem da stabilni sustav, s novim funkcionalnostima može biti puštan u rad u sukcesivnim iteracijama putem primjene mikroservisnih principa. Mikroservis je nezavisno isporučiva komponenta Rješenja koji podržava interoperabilnost putem komunikacije porukama. Mikroservisna arhitektura je stil programskog inženjerstva u kojem su visoko automatizirani i nadogradivi softverski sustavi sastavljeni od podjednako sposobnih mikroservisa.

Ključna obilježja koja opisuju mikroservise su sljedeća:

- male veličine (termin „mikro“ ne odnosi se nužno na veličinu servisa u pogledu linija koda, već na njegovu funkcionalnost - svaki mikroservis slijedi princip jedne odgovornosti (engl. *Single Responsibility Principle*)),
- komunikacija porukama (komunikacija između mikroservisa najčešće se realizira putem HTTP protokola, koristeći RESTful API sučelja),
- vezani kontekstom (mikroservisi su visoko kohezivni te rade zajedno u sklopu jednog sustava),
- samostalan razvoj (svaki mikroservis se može razvijati neovisno o ostalima) te
- nezavisna isporuka (svaki mikroservis se može isporučiti neovisno o ostalima).

Svi mikroservisi unutar Rješenja moraju biti izolirana okruženja (koja uključuju svoje binarne (izvršne) datoteke i biblioteke), unutar jednog operativnog sustava (engl. *host*), kako je prikazano na slici u nastavku.



Slika 1: Mikroservisna arhitektura Rješenja

3. Sustav za kontejnerizaciju

Kako bi se postigla željena mikroservisna arhitektura (kako je opisano u prethodnom odjeljku), potrebno je koristiti kontejnere.

Sve datoteke potrebne za pokretanje mikroservisa sadržane su u kontejneru. Kontejner može imati zasebna mrežna sučelja koja se mogu razlikovati od onih na *host-u* što bi inače moglo dovesti do konflikata pri korištenju pojedinih brojeva portova.

4. Operativni sustav

Naručitelj koristi i ima operativno znanje i iskustvo u upravljanju Debian operativnim sustavom otvorenog koda te rješenje mora podržavati mogućnost implementacije na operativnom sustavu otvorenog koda, korištenjem kojeg naručitelj neće snositi dodatne licenčne, operativne ili resursne troškove.

5. Infrastruktura

Rješenje je potrebno u potpunosti primijeniti u okolini naručitelja (engl. *On-Premise*).

Rješenje je potrebno realizirati s posebnim naglaskom na pouzdanost, skalabilnost, visoku dostupnost i sigurnost.

Naručitelj će za potrebe Rješenja osigurati testnu i produkcijsku kontejner platformu. Razvojnu platformu usporedivih funkcionalnosti dužan je osigurati odabrani ponuditelj.

Naručiteljeva infrastruktura je zasnovana na OKD4 platformi (engl. *the community distribution of Red Hat OpenShift Kubernetes platform*) te traženo rješenje treba biti prilagođeno korištenju na OKD4 platformi.

6. API menadžment

Naručitelj za razvoj Rješenja zasnovanih na mrežnim (*eng. web*) aplikacijama koristi API sučelja i pozive koje klijenti mogu koristiti za interakciju s aplikacijom.

API Rješenja treba podržavati:

- **neovisnost platforme**
  - svaki klijent treba biti u mogućnosti pozvati API, bez obzira na to kako je API interno implementiran. Navedeno zahtijeva korištenje standardnih protokola i uspostavljanje mehanizma kojim klijent i web usluga definiraju format podataka za razmjenu
- **razvoj usluge**
  - web API treba imati mogućnost razvoja i dodavanja funkcionalnosti neovisno o klijentskim aplikacijama. Kako se API razvija, postojeće klijentske aplikacije trebaju biti u mogućnosti nastaviti funkcionirati bez izmjena. Sve funkcionalnosti trebaju biti dostupne na način da ih klijentske aplikacije mogu u potpunosti koristiti.

6.1. REST API

REST implementacije najčešće koriste HTTP kao aplikacijski protokol, a ponuđeno Rješenje se mora temeljiti na REST API-jima izgrađenim na HTTP-u kako bi se moglo koristiti i povezivati s postojećim sustavim i rješenjima naručitelja.

U nastavku su raspisani specifični zahtjevi REST API-ja koje Rješenje mora ili treba zadovoljavati. Ukoliko je naznačeno da „mora“ zadovoljavati, zahtjev je obavezan. U suprotnom, zahtjevi nisu obavezni.

PODRUČJA	SPECIFIČNI ZAHTJEVI
Korištenje HTTP naredbi	Ispravne HTTP naredbe (GET, PUT, POST, DELETE, OPTIONS, HEAD, PATCH) moraju se mapirati za obavljanje operacija na REST resursu.
Pravila imenovanja	- imena resursa moraju uvijek biti u množini, - treba koristiti imenice za imena resursa,

REST API resursa	- treba koristiti simbol (-) ako resurs ima više riječi (npr. „/imena-ucenika“) te - resurs treba imati konkretno ime.
Pravila korištenja jedinstvenog identifikatora resursa (engl. *Uniform Resource Identifier*, URI)	- URI mora biti napisan malim slovima, - mora se koristiti simbol (-) ukoliko URI ima više riječi (engl. *kebab-case*), - treba koristiti što je moguće kraći path („api/v1/učenici“ radije nego „api/v1/škola/5-razred/c/učenici“), - ne smije se sve *cross*-referencirati te - ne smiju se koristiti znakovi (/) i (_).
Pravila uz verzije REST API-ja	- verzija API-ja mora biti navedena u URI-ju, - verzija API-ja mora uvijek biti prisutna (tj. uključujući i prvu verziju), - verzija API-ja mora biti navedena kao "v", - ako korisnik pokuša doći do API-ja bez navođenja verzije, mora dobiti odgovor 404 pogreške, - nova verzija API-ja mora se napraviti ako su postojeći resursi modificirani na način koji utječe na korisnike (npr. novo obavezno polje, brisanje polja i sl.) - nova verzija API-ja mora se napraviti ako se naprave nove vrste pogrešaka, - nova verzija API-ja mora se napraviti ako se naprave novi resursi s kojima klijenti moraju komunicirati i sl.
Pravila vezana uz ne-RESTful operacije na resursu	- broj akcija mora se nastojati minimalizirati, - treba koristiti glagole za imenovanje akcija, - akcije treba izolirati od ostatka API-ja pod resursom „akcije“ te - ako se akcija mora definirati odvojeno od stvarnih resursa, onda se mora osigurati da je jasno dokumentirano da akcija nije resurs.
Pravila dokumentiranja REST API-ja	- treba biti lako dostupna i jednostavna za pretraživanje, - treba sadržavati primjere cjelovitih zahtjeva / odgovora te - treba sadržavati obavijesti o životnom ciklusu i prestanku upotrebe API-ja (engl. *API depreciation*).
Sigurnosna pravila REST API-ja	- svi ulazi i izlazi (engl. *inputs and outputs*) moraju se ispravno provjeriti i kodirati, - sve se mora poslužiti preko sigurnih kanala, koristeći TLS protokol (TLS 1.2 i novije verzije), - ne smiju se slati tragovi stoga (engl. *stack traces*) API klijentima (5xx), - REST API treba podržavati CORS samo ako mu treba pristupiti s različitih domena, - moraju se izostaviti sva sigurnosno osjetljiva zaglavlja (npr. otkrivanje verzija poslužitelja / biblioteka), - sav loš unos od strane klijenta (4xx kodovi) se mora odbiti, - sve sumnjive aktivnosti se moraju evidentirati, - mora se uspostaviti nadzor (engl. monitoring) radi otkrivanja abnormalnog korištenja API-ja,

	- korištenje API-ja treba kontrolirati i regulirati kako bi se ublažili napadi uskraćivanja usluge (engl. *Denial of Service*, DoS) i blokirali zlonamjerni korisnici i sl.
Paginacija	- zahtjevi za zbirkama uvijek moraju biti paginirani, - API klijenti moraju biti u mogućnosti jasno odrediti odsječak skupa podataka koji žele dohvatiti (moraju koristiti atribute ograničenja i *offset* niza upita (engl. *query string*) za paginaciju - ograničenje i *offset* mora odrediti API klijent. Ukoliko API klijent nije odredio atribute, moraju se primijeniti zadane vrijednosti, ograničenje (10), *offset*: (0) te - paginirani odgovor mora sadržavati: niz "stavki" koji sadrži odgovarajuće rezultate i objekt "metapodataka“.
Filtriranje	- označavanje polja koja se žele filtrirati treba biti zadani pristup filtriranju u sklopu Rješenja.
Sortiranje	- treba upotrijebiti parametar niza upita sa sljedećom sintaksom: “sort=<field1>+<ASC|DESC>[,<field2>+<ASC|DESC>][, …]“. - prilikom sortiranja „metapodataka“, objekt „metapodataka“ mora sadržavati sortedBy, a sortedBy mora sadržati polje i redoslijed (ASC/DESC).
Grupne operacije	- Rješenje treba podržavati sljedeće grupne operacije: grupno kreiranje, grupno ažuriranje te grupne operacije koje ujedno kreiraju i ažuriraju. Grupno brisanje ne treba biti podržano osim u jasno definiranim slučajevima.
Vrste medija	- vrsta medija mora biti navedena u sljedećim zaglavljima: - zahtjevi („Accept: <media type(s)>“ - očekivana vrsta medija odgovora) te - odgovori (vrsta medija), - API mora poslati kod pogreške 406 ako ne može generirati nijednu preferiranu vrstu medija zatraženu od strane klijenta te - API mora poslati kod pogreške 415 ako ne može obraditi vrstu medija koju isporučuje klijent.
Pravila predmemoriranja	- Rješenje treba osigurati da svaki odgovor poslužitelja daje ispravne HTTP smjernice zaglavlja kako bi uputio preglednik kada i koliko dugo preglednik može predmemorirati odgovor.
Veličina datoteke za učitavanje	- ako vatrozid web aplikacije (engl. *Web Application Firewall*, WAF) ispred API-ja treba pregledati datoteke, tada datoteke ne bi trebale biti veće od 20 MB te - za veće datoteke (> 100MB), treba razmotriti korištenje SFTP-a (engl. *SSH File Transfer Protocol*).

7. Baza podataka

Baza podataka je skup međusobno povezanih podataka pohranjenih na vanjskoj memoriji, koji su istovremeno dostupni raznim korisnicima i aplikacijama. Upisivanje, promjena, brisanje i čitanje podataka obavlja se posredstvom sustava za upravljanje bazom podataka (engl. *Data Base Management System - DBMS*), a korisnici i aplikacije pritom ne moraju poznavati detalje fizičkog prikaza podataka, već se referenciraju na logičku strukturu baze.

DBMS je poslužitelj baze podataka koji obavlja u ime klijenata sve operacije s podacima, podržava više baza podataka svaka sa svojom logičkom strukturom i s istim modelom, brine se za sigurnost podataka, te automatizira administrativne poslove s bazom.

Rješenje mora podržavati besplatan *open source* DBMS koji ima mogućnost svakom korisničkom računu poslužitelju dodijeliti upravljačka prava na cijeli poslužitelj ili pojedine baze, a čijim pravima upravlja superadministrator od strane naručitelja.

Dodatno, podržani DBMS mora zadovoljavati sljedeće uvjete:

- kompatibilan za web aplikacije i web stranice (uključujući odabrani CMS),
- mogućnost jednostavnog brisanja podataka (npr. osobnih podataka pojedinca) bez narušavanja integriteta baze podataka,
- mogućnost enkripcije i pseudonimizacije osobnih podataka (odnosi se na „maskiranje“ podataka),
- mogućnost izvoza osobnih podataka u opće prihvaćenom elektroničkom obliku,
- dokumentiranje svih promjena na bazi podataka (tko, kada i gdje je napravio promjene),
- mogućnost obavještavanja o pokušaju neovlaštenog pristupa / unosa promjena na bazu podataka,
- mogućnost postavljanja razine sigurnosti zaporke instalacije,
- mogućnost definiranja lozinke za korijenskog korisnika (engl. *root user*),
- mogućnost uklanjanja anonimnih računa,
- mogućnost uklanjanja testnih baza podataka koje su, prema zadanim postavkama, dostupne svim korisnicima,
- mogućnost automatske izrade sigurnosne kopije prema definiranim intervalima,
- mogućnost repliciranja baze podataka

Naručitelj za DBMS koristi i ima operativno znanje i iskustvo sa sljedećim poslužiteljima baza podataka baziranih na otvorenom kodu.

- PostgreSQL
- MariaDB

Rješenje mora podržavati mogućnost implementacije na poslužiteljima baza podataka baziranih na otvorenom kodu korištenjem kojih naručitelj neće snositi dodatne licenčne, operativne ili resursne troškove.

8. Poslužitelj mrežnih stranica i aplikacija

Poslužitelj mrežnih stranica i aplikacija (engl. *web server*) omogućuju komunikaciju između klijenta (npr. uređaja krajnjih korisnika, učenika, nastavnika, itd.) i aplikacije unutar Rješenja putem Interneta. Sadržaj iz baze podataka poslužitelj mrežnih stranica i aplikacija povlači na svaki zahtjev klijenta i dostavlja ga u web preglednik klijenta. Poslužitelj istovremeno poslužuje više različitih web klijenata s različitim zahtjevima. Web poslužitelji izvršavaju programski kod, pretvara ih u statičke HTML datoteke i poslužuje ih u web pregledniku klijenta.

Web server u sklopu Rješenja mora minimalno zadovoljavati sljedeće uvjete:

- Poslužitelj mrežnih stranica i aplikacija baziran na softveru otvorenog koda
- Više platformski (engl. *cross - platform*) softver koji podržava različite operativne sustave,
- Komunikacija klijent - poslužitelj preko HTTP protokola,
- Modularna struktura koja omogućava administratorima da zasebno upravljaju funkcionalnostima web servera (sigurnosne postavke, predmemorija, URL promjene (engl. *rewriting*), provjera autentičnosti lozinke i sl.),
- kompatibilnost s web aplikacijama i web stranicama (uključujući odabrani CMS), podrška za programske jezike na strani servera,
- podrška za TLS / SSL,
- mogućnost virtualnog udomljavanja aplikacija,
- mogućnost korištenja više različitih web servera istovremeno (npr. Apache i Nginx),
- mogućnost automatskog skaliranja poslužitelja do istovremenog posluživanja minimalno 20% potencijalnih korisnika rješenja u minuti
- dostupna podrška putem zajednice ili više lokalnih partnera

Naručitelj za poslužitelj mrežnih stranica i aplikacija koristi i ima operativno znanje i iskustvo sa sljedećim poslužiteljima mrežnih stranica i aplikacija baziranih na otvorenom kodu.

- Apache
- Nginx

Naručitelj održava web aplikacije napisane u programskom kodu PHP.

Rješenje mora podržavati mogućnost implementacije na poslužiteljima mrežnih stranica i aplikacija baziranih na otvorenom kodu korištenjem kojih naručitelj neće snositi dodatne licenčne, operativne ili resursne troškove za udomljavanja ili podršku.

9. Programski kod i programski jezici

9.1. Vlasništvo nad izvornim kodom

Vlasništvo nad izvornim kodom te pripadajućom razvojnom, tehničkom i korisničkom dokumentacijom sa svim traženim funkcionalnostima odabrani ponuditelj je obvezan, tijekom primopredaje sustava osnovati u korist naručitelja, uključujući isključivo pravo iskorištavanja koje je sadržajno, prostorno i vremenski neograničeno. Ugovorom o javnoj nabavi odabrani ponuditelj jamči naručitelju da naručitelj potpisom ugovora o javnoj nabavi stječe sva imovinska prava kao i pravo osnivanja daljnjeg prava iskorištavanja te da je autor na navedeno dao svoju pisanu suglasnost. Autor zadržava pravo iskorištavanja za sebe i odabrani ponuditelj se obvezuje urediti sve odnose s autorima i nositeljima autorskih prava vezanih za ovaj predmet nabave.

Odabrani ponuditelj se obvezuje naručitelju predati u posjed izvorni kod programskog rješenja i sve potrebne programske biblioteke programske platforme za razvoj web aplikacija u strojnom kodu i pripadajuću dokumentaciju i time prenijeti na naručitelja pravo modifikacije i daljnjega razvoja programskog rješenja.

Naručitelj može koristiti izvorni kod programskog rješenja za razvoj drugih aplikacija. Odabrani ponuditelj zadržava pravo korištenja izvornog koda te ima pravo isti dati i trećim osobama.

9.2. Isporuka izvornog koda

Za traženo Rješenje odabrani ponuditelj mora osigurati uvid u programski kod te da se izvorni programski kod Rješenja pohrani na predviđeno mjesto repozitorija koda naručitelja. Odabrani ponuditelj mora osigurati i dokumentirati mogućnost promjene koda Rješenja prema potrebama i na zahtjev naručitelja. Odabrani ponuditelj je dužan osigurati da se sve promjene na produkcijskoj verziji Rješenja šalju isključivo s naručiteljevog repozitorija koda.

- Naručitelj osigurava testnu i produkciju okolinu (OKD4 platforma)
- Ponuditelj se upoznaje s naručiteljevom platformom i odgovoran je za izradu i provedbu build i deploy procesa na naručiteljevoj platformi na testnom okruženju.
- Naručitelj prebacuje s testa na produkciju i odgovoran je za build i deploy na produkcijskom dijelu u suradnji sa ponuditeljem

9.3. Dokumentacija

Odabrani ponuditelj je obavezan predati naručitelju dokumentaciju koja uključuje:

- funkcionalnosti sustava - popis ključnih funkcionalnosti i namjena (opis funkcionalnosti)

- nefunkcionalni opis sustava - računalna platforma (klijentska razina, poslužiteljska razina), performanse i raspoloživost, sigurnost, komunikacija s vanjskim sustavima, dizajn sučelja, katalog stilova i slično
- arhitekturu sustava i opis modela podataka - aplikativna arhitektura sustava (platforma, klijentska razina - prezentacijski i servisni sloj, poslužiteljska razina - servisni i podatkovni sloj), izvedbena arhitektura sustava (servisni i podatkovni sloj - broj poslužitelja, VM, fizički poslužitelji, zaštita sustava, međusobna povezanost, ...), okoline i instance sustava (razvojna, testna i produkcijska okolina)
- korištene tehnologije i razvojni alati - popis tehnologija i alata te vrsta i namjena
- upute za korisnike - upute za različite profile korisnika
- dokumentaciju koda s konvencijom imenovanja
- postupke testiranja, instalacije i nadogradnje sustava
- dokumentaciju API dijela sustava
- prijedlog testnih scenarija za testiranja u kojima sudjeluje naručitelj.

Dokumentaciju će ponuditelj izraditi u dogovoru s naručiteljem.

10. Testiranje

Minimalna testiranja koja je potrebno provoditi odnose se na sljedeće:

- integracijski testovi (engl. Integration Testing): Integracijsko testiranje je faza testiranja u sklopu koje se pojedinačni moduli i/ili programske jedinice spajaju i testiraju zajedno kao cjelina. Integracijsko testiranje provodi se po završetku jediničnog testiranja, a za cilj ima provjeriti funkcionalnost sučelja među komponentama (modulima i/ili programskim jedinicama) – razotkrivanje nedostataka sučelja i interakcije između integriranih komponenti. Integracija se provodi postupno, na način da se uključuju komponente za koje je prethodno utvrđeno da ispravno rade. U navedenom trenutku, neke druge komponente mogu biti u fazi razvoja i/ili jediničnog testiranja. Integracijsko testiranje će u konačnici osigurati da sve komponente ispravno rade i nakon integracije s ostalim komponentama,
- performansni testovi (engl. Performance Testing): Performansni testovi se provode nad infrastrukturnim segmentom koji podržava rad planiranog IT rješenja pri čemu je cilj simulirati velik broj istovremenih prijava/akcija na planiranom IT rješenju kako bi se utvrdile performanse navedenog IT rješenja te identificirale eventualne manjkavosti (navedenim testiranjem se provjerava valjanost drugih kvalitativnih osobina sustava, kao što su skalabilnost, pouzdanost i korištenje resursa). Performansi testovi mogu uključivati i testove izdržljivosti,
- sigurnosna testiranja (engl. Security Testing): SAST (engl. Static Application Security Testing) testiranje na programskom kodu te DAST (engl. Dynamic Application Security Testing) i IAST (engl. Interactive Application Security Testing) testiranje na dijelovima planiranog IT rješenja tijekom njihovog razvoja.

Detaljnije o usklađenosti sustava sa sigurnosnim standardima i preporukama u posebnom poglavlju,

- regresijska testiranja (engl. Regression Testing): Regresijsko testiranje je fokusirano na pronalaženje greški, koje se pojavljuju nakon promjena programskog koda. Takve regresije događaju se kada god funkcionalnosti IT rješenja koja su prethodno ispravno radile prestaju raditi onako kako je inicijalno zamišljeno te
- korisnička testiranja (engl. User Acceptance Testing): Provjera ili testiranje prihvatljivosti je testiranje koje provode poslovni korisnici u trenutku kada je IT rješenje spremno za isporuku, a nakon što se ispravila većina grešaka identificirana u ranijim fazama testiranja. Općenito, to je završni korak u cjelokupnom procesu testiranja pri čemu se provjeravaju funkcionalnosti prema ranije definiranim specifikacijama te se određuje da li IT rješenje ispunjava potrebe krajnjih korisnika. Navedeno testiranje provode krajnji korisnici IT rješenja kako bi se osiguralo da navedeno rješenje omogućuje sve funkcionalnosti kako je to inicijalno bilo predviđeno.

Integracijski, performansni i regresijski testovi su obaveza odabranog ponuditelja.

Sigurnosna i testiranja korisničkog prihvaćanja odrađuje Naručitelj u suradnji sa odabranim ponuditeljem.

11. Usklađenost sa smjernicama za osiguravanje digitalne pristupačnosti

Sukladno Zakonu o pristupačnosti mrežnih stranica i programskih rješenja za pokretne uređaje tijela javnog sektora Republike Hrvatske (NN 17/19), na snazi od 23. rujna 2019., te sukladno Direktivi (EU) 2016/2102 Europskog parlamenta i Vijeća od 26. listopada 2016. o pristupačnosti internetskih stranica i mobilnih aplikacija tijela javnog sektora (SL L 327, 2.12.2016, str. 1.), mrežne stranice i programsko rješenje treba biti izrađeno tako da su dizajn, funkcionalnosti i sam sadržaj pristupačni svim korisnicima, uključujući osobe s invaliditetom (oštećenja vida, sluha, govora, motorike, kognitivna i neurološka oštećenja) i osobe koje koriste pomoćnu tehnologiju, u mjeri u kojoj to tehnologija kojom je kreirano omogućuje.

Odabrani ponuditelj/izvršitelj dužan je predmet nabave izraditi u skladu s poglavljem 7 CARNET-ovih Smjernica za osiguravanje digitalne pristupačnosti, dostupnih putem poveznice www.carnet.hr/pristupacnost, čime će zadovoljiti nužni minimum za osiguravanje pristupačnosti mrežnih stranica i programskih rješenja za pokretne uređaje.

Kada je to moguće, odabrani ponuditelj/izvršitelj treba primijeniti opširnije W3C WAI standarde za izradu pristupačnih sadržaja, mobilnih sadržaja i aplikacija (Web Content Accessibility Guidelines (WCAG) 2.1 i Rich Internet Applications (WAI – ARIA) 1.1).

Proces provjere digitalne pristupačnosti

Rezultati koje isporučuje odabrani ponuditelj bit će podvrgnuti testiranju od strane naručitelja radi provjere njihove pristupačnosti za osobe s invaliditetom i osobe koje koriste pomoćnu tehnologiju.

Postupak provjere digitalne pristupačnosti naručitelj će provesti nakon izrade i isporuke rezultata od strane odabranog ponuditelja/izvršitelja.

Rezultate testiranja digitalne pristupačnosti naručitelj će dostaviti odabranom ponuditelju/izvršitelju u pisanom obliku u roku od mjesec dana od početka testiranja. Odabrani ponuditelj dužan je unijeti ispravke i dorade mrežnih stranica i programskog rješenja za pokretne uređaje za sve slučajeve za koje rezultati testiranja pokažu da je to neophodno, kako bi se osigurala jednaka mogućnost pristupa i korištenja svim korisnicima.

Pri planiranju prilagodbe mrežnih stranica i programskog rješenja, odabrani ponuditelj/izvršitelj treba predvidjeti vrijeme potrebno za provođenje testiranja digitalne pristupačnosti od strane naručitelja. Pritom odabrani ponuditelj/izvršitelj treba uzeti u obzir mogućnost potrebe za ispravkom i doradom mrežnih stranica i programskog rješenja temeljem rezultata testiranja i planirati vrijeme potrebno za izmjene i dorade.

12. Usklađenost sustava sa sigurnosnim standardima i preporukama

Sigurnost po dizajnu

Ponuditelj treba osigurati da je sustav i svi njegovi elementi u skladu s najboljom sigurnosnom praksom[1],tako da štiti sebe i informacije koje obrađuje te pruža otpor napadačima.

Ako prilagodbe sustava za potrebe naručitelja budu uključivale integraciju ili povezivanje s drugim naručiteljevim sustavima ili uslugama, one također moraju biti izrađene u skladu s najboljom sigurnosnom praksom, prema principima sigurnosnog programiranja i takve da je sigurnost ugrađena u njih od samog dizajna sustava.

Sigurnosno testiranje sustava

Nakon izrade svih funkcionalnosti sustava, a prije stavljanja u produkcijsko okruženje odabrani ponuditelj dužan je omogućiti naručitelju provođenje sigurnosnog testiranja. Postupak sigurnosnog testiranja podrazumijeva detekciju eventualnih sigurnosnih propusta u sustavu automatiziranom analizom korištenjem specijaliziranih alata te ručne provjere sigurnosti. Opseg sigurnosnog testiranja opisan je u Poglavlju: "***Opseg sigurnosnog testiranja sustava***"

Sigurnosno testiranje obavljat će naručitelj, a odabrani ponuditelj dužan je omogućiti provođenje testiranja od strane naručitelja. Odabrani ponuditelj obvezan je otkloniti eventualne sigurnosne propuste sukladno rezultatima testiranja.

---

1 Na primjer, OWASP Application Security Verification Standard 4.0 ili jednakovrijedno – poželjno Level 3, (https://github.com/OWASP/ASVS/blob/master/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0-en.pdf)

Postupak sigurnosnog ispitivanja podrazumijeva detekciju eventualnih sigurnosnih propusta u aplikaciji automatiziranom analizom korištenjem specijaliziranih alata te ručne provjere sigurnosti.

Osim prije inicijalnog postavljanja sustava u produkcijsku okolinu, testiranje će se provoditi: periodički (najviše jednom godišnje za vrijeme trajanja ugovora) te izvanredno (na zahtjev naručitelja, u slučaju veće nadogradnje sustava ili u slučaju sigurnosnog incidenta).

Uklanjanje uočenih sigurnosnih propusta

Razvijeni programski sustav se ne smije staviti u produkciju ako nisu ispravljeni pronađeni propusti označeni kao kritični ili srednji. Nakon što odabrani ponuditelj obavijesti naručitelja da je ispravio pronađene propuste koji stvaraju prepreku za puštanje sustava u produkciju biti će provedena dodatna sigurnosna provjera sustava. Dodatna sigurnosna testiranja ponavljaju se sve dok sustav nije spreman za produkciju.

U slučaju periodičkih ili izvanrednih sigurnosnih testiranja odabrani ponuditelj je dužan ispraviti uočene sigurnosne propuste prema rokovima definiranima Tablicom:

Rokovi za uklanjanje sigurnosnih propusta

Razina kritičnosti sigurnosnog propusta[2]	**Rok za uklanjanje sigurnosnog propusta**
Sigurnosni propusti kritične razine	do 3 dana od zaprimanja rezultata testiranja
Sigurnosni propusti srednje razine	do 7 dana od zaprimanja rezultata testiranja
Sigurnosni propusti niske razine	do 30 dana od zaprimanja rezultata testiranja

12.1. Opseg sigurnosnog testiranja sustava

Elementi provjere sigurnosti:

1. Udaljeno ispitivanje aplikacije

Udaljeno ispitivanje aplikacije obuhvaća ispitivanje aplikacije iz korisničkog pogleda s ciljem detektiranja mogućnosti izvođenja neovlaštenih korisničkih aktivnosti. Ovo ispitivanje podrazumijeva korištenje skenera te izvođenje ručnih provjera.

2. Lokalno ispitivanje samostalno razvijene aplikacije

Ispitivanje obuhvaća ispitivanja navedena u točki 1. uz ispitivanje izvornog koda aplikacija koji je samostalno razvijen i vanjskih gotovih modula ako se koriste. Ispitivanje obuhvaća i ispitivanje konfiguracijskih postavki aplikacije.

[2] Razina kritičnosti sigurnosnog propusta kako je u izvještaju o testiranju definirao Izvođač

13. Jamstveni rok

Za vrijeme trajanja jamstvenog roka Odabrani ponuditelj je dužan poduzeti sve radnje i popravke, uključivo nužnu i sigurnosnu nadogradnju sustava (uključujući softver bilo koje komponente sustava) koje su potrebne da bi se otklonili nedostatci u funkcioniranju sustava te sigurnosne ranjivosti i sigurnosne propuste. Radnje koje poduzima Odabrani ponuditelj za vrijeme jamstvenog roka odnose se isključivo na otklanjanje nedostataka, sigurnosnih ranjivosti i sigurnosnih propusta, te neispravnost u radu sustava za vrijeme jamstvenog roka.

Vremensko trajanje jamstvenog roka je 5 godina.