



Prolećni semestar 2019/2020

Goose Deployment System

SE201

Uvod u softversko inženjerstvo

Projektna dokumentacija

Student:

Nikola Tasić 3698

Profesor:

Dragan Domazet

Asistent:

Jovana Jović

Sadržaj

| | |
|---|----|
| Uvod..... | 4 |
| Svrha dokumenta | 4 |
| Terminologija | 4 |
| Opis | 5 |
| Funkcije softvera | 5 |
| Korisnici sistema | 5 |
| Opis sistema za korisnika | 5 |
| Metodologija izrade | 6 |
| Specifikacija zahteva | 6 |
| Funkcionalni i nefunkcionalni zahtevi | 6 |
| Slučajevi upotrebe | 9 |
| Slučaj upotrebe postavljanja aplikacije..... | 9 |
| Slučaj upotrebe ažuriranja aplikacije | 10 |
| Slučaj upotrebe brisanja aplikacije | 11 |
| Slučaj upotrebe promene parametara aplikacije | 12 |
| Dijagrami sekvenci | 13 |
| Dijagram postavljanja aplikacije | 13 |
| Dijagram ažuriranja aplikacije | 14 |
| Dijagram brisanja aplikacije | 15 |
| Dijagram promene parametara aplikacije | 15 |
| Deteljan dijagram postavljanja aplikacije | 16 |
| Klasni dijagram | 16 |
| Arhitektura sistema..... | 17 |
| Opis arhitekture sistema..... | 17 |
| Korisnički interfejs..... | 18 |
| Login forma | 18 |
| Pregled postavljenih aplikacija..... | 18 |
| Pregled informacija o pokrenutoj aplikaciji | 19 |
| Deployment interfejs | 21 |

| | |
|------------------|----|
| Testiranje | 22 |
| Zaključak | 22 |

Uvod

Ovaj dokument služi da bolje opiše funkcionalnosti proces izrade **Goose** sistema za deployment aplikacija. Ovaj projektni zadatak baviće se definisanjem osnovnih funkcionalnih i nefunkcionalnih zahteva koje softver mora da zadovolji kako bi se smatrao uspešnim. U dokumentu će biti detaljno opisan rad sistema o kome je reč. Za bolje razumevanje problema sa kojim se susrećemo kako bi projekat bio realizovan biće objašnjeni putem korišćenja dijagrama (Klasni, UseCase, sekvencijalni) koji bi trebalo detaljnije da objasne način i zamisao funkcionisanja sistema. Za potrebe izrade dijagrama koristiće se alata PowerDesigner. Prikaz interfejsa za korišćenje biće realna aplikacija koja već radi u produkciji i pokazuje rezultate. Korisnici ovog sistema su pojedinačni developeri ili jednostavno ljudi koji žele da imaju olakšani pregled i upravljanje svojim web aplikacijama.

Projekat je tako zamišljen da olakša proces postavljanja aplikacija na web servere, njihovo ažuriranje i da omogući lak način za održavanje više web aplikacija na jednom domenu i jednom fizičkom serveru.

Svrha dokumenta

Svrha dokumenta je da jasno opiše moguće funkcionalnosti i mogućnosti softvera. Dokument se sastoji od samog uvoda gde se daje kratak opis cilja za kreiranje ovog sistema. Nakon toga, dokument će se sastojati od opisa sistema njegovim mogućnostima, detalje zahteva u kojima su opisani funkcionalni i nefunkcionalni zahtevi. Dokument će se sastojati i od specifikacije dizajna u okviru kog se nalaze dijagrami slučajeva korišćenja, klasni dijagram i svih mogućih sekvencijalnih dijagrama koji su vezani za rad sistema za praćenje prisustva studenata. Nakon toga, dokument će dati opis funkcionisanja sistema u vidu slika i sam dizajn tog sistema u realnoj situaciji.

Terminologija

Kako će se u dokumentu koristiti terminologija koja zavisi od konteksta i profesionalna terminologija prilikom objašnjavanja i razvoja sistema, potrebno je da čitaoc ovog dokumenta upoznamo sa skraćenicama i definicijama koje ćemo koristiti u razvijanju dokumenta.

- **Goose** – naziv sistema.
- **PowerDesigner** – Grafičko okruženje za modelovanje aplikacija, podataka, poslovnih procesa.
- **Web server** – Fizička mašina koja je povezana na internet i ima svoju internet adresu.
- **Aplikacija** – Ova reč će se koristiti u kontekstu web aplikacije koju korisnik želi da postavi (deploy) na web server na kome radi Goose sistem.
- **Deployment** – proces koji počinje prebacivanjem binarnih fajlova ili izvornog koda aplikacije na web server a završava se aktivnom aplikacijom na istom.
- **YAML** – struktuirani jezik za opisivanje i čuvanje podataka.
- **JSON** – struktuirani jezik za opisivanje i čuvanje podataka.

- **Go** – programski jezik u kome je razvijen backend aplikacije.
- **React** – tehnologija za izradu web stranica i interfejsa
- **Kontejner** – virtuelizovano okruženje u kome proces tj. Web aplikacija radi.
- **Routing** – preusmeravanja zahveva na aplikacije u zavisnosti od adrese koja je zahtevana.
- **Backend** – Logika aplikacije koja se izvršava na web serveru.
- **Log** – tekstualni fajl sa opisom proteklih dešavanja u procesu.
- **Terminal, Shell** – tekstualni korisnicki interfejs
- **SSH** – protokol sigurnog upravljanja tekstualnih korisničkih interfejsa preko mreže
- **MVP** – minimum viable product
- **API** - interfejs za upravljanje aplikacijama

Opis

Goose je sistem dizajniran da olaksa proces deploymenta aplikacija na server i upravljanje vise različitih web aplikacija na jednoj fizičkoj ili virtuelnoj mašini. Goose omogućava developšerima brz i lak način testiranja svojih web aplikacija na realnom hardveru ali takođe s druge strane omogućava platformu sa pokretanje više različitih sistema koji mogu da međusobno komuniciraju. Svaka aplikacija je bezbedna i radi u okviru svog kontejnera i u slučaju bilo kakvih katastrofalnih grešaka oprativni sistem koji sve to pokreće ostaje zaštićen.

Funkcije softvera

- **Autentikacija korisnika sistema**
- **Deployment aplikacija na web server**
- **Podešavanje routinga aplikacija**
- **Ažuriranje aplikacija**
- **Brisanje aplikacija**
- **Konfiguracija parametara postavljenih aplikacija**
- **Pregled logova aplikacija i samog sistema**

Korisnici sistema

Postoji samo jedan tip korisnika a to je sam developer koji koristi ovaj sistem.

Opis sistema za korisnika

Korisnik sistema treba da samo pokrene binarnu distribuciju ovog sistema u terminalu i dobiće funkcionalan sistem sa onim sto u struci zovemo *reasonable defaults* odnosno podrazumeavim podešavanjima koja su validna i omogućavaju rad sistema a ne utiču na ostale već postojeće programe. Korisniku se u terminalu ispisuju sva trenutna podešavanja i informacije kako može da pristupi

korisnijem web interfejsu preko svog pretraživača. Posle uspešne autentikacije korisnik ima pristup interfejsu gde odmah može krenuti sa deployovanjem aplikacija. Konfiguracija sistema se vrši preko YAML fajlova za koje korisnik u terminalu dobija informaciju gde se nalaze i kako treba izmeniti.

Metodologija izrade

Metodologija koju ćemo koristiti za izradu ovog sistema je **agilni metod razvoja**. Za izbor ovog metoda je zaslužna činjenica da se ovaj sistem sa stoji iz više komponenti koje mogu biti razvijene nezavisno jedna od druge. Backend ovog sistema je servisno orijentisan web server koji komunicira preko JSON HTTP zahteva što znači da korisnički interfejs za čiju bazu je odabran React može biti razvijan paralelno jer su interfejsi komunikacije između ova dva sistema već jasno predefinisani u dokumentu. Takođe sistem kontejnerizacije aplikacija na serveru je podsistem koji je na primer takođe autnoman i može se razvijati nezavisno. Agilnim metodom možemo vooma brzo doći do MVP-a na primer backenda koji će biti funkcionalan bez obzira na to kakav korisnički interfejs komunicira sa njim. Svaki novi inkrement može da dodda novu funkcionalnst ili celu novu komponentu sistema.

Specifikacija zahteva

Funkcionalni i nefunkcionalni zahtevi

| Zahtev | Opis |
|--------|---|
| 1.1 | Konfiguracija sistema Ulogovani korisnik ima porstup interfejsu za konfiguraciju sistema |
| 1.1.1 | Osnovna konfiguracija Korisnik u konfiguracionom interfejsu ima pristup izmeni parametara vezanih za, Goose proxy, Goose deployer i izmenu korisnickog imena i sifre. |
| 1.1.2 | Obavestjenje o izmeni Korisnik dobija obavestjenja da li su nova podesavanja validna i uspesno sacuvana. |
| 1.2 | Postavljanje aplikacija Ulogovani korisnik ima pristup interfejsu za postavljanje aplikacija |
| 1.2.1 | Interfejs za postavljanje Interfejs za postavljanje sadrzi polja za unos linka do repozitorijuma aplikacije, |

| | |
|-------|---|
| | <p>bekend tehnologiju koja ce pokretati tu aplikaciju na Goose serveru i hostname koji omogucava pristup aplikaciji preko web-a.</p> |
| 1.2.2 | <p>Obavestenje o postavljanju</p> <p>Korisnik dobija obavestenje o uspesnosti postavljanja aplikacije na server i mogucim greskama koje su dosle u toku posavljanja.</p> |
| 1.3 | <p>Konfiguracija aplikacija</p> <p>Korisnik ima pristup interfejsu za izmenu konfiguracionih parametara za svaku postavljenu aplikaciju</p> |
| 1.3.1 | <p>Interfejs za konfiguraciju</p> <p>Interfejs za konfiguraciju sadrzi unosna polja za promenu parametara koji su bili opcije kod postavljanja aplikacije.</p> |
| 1.3.2 | <p>Obavestenje o izmeni</p> <p>Korisnik dobija obavestenje o uspenosti izmene konfiguracionih parametara aplikacije</p> |
| 1.4 | <p>Pregled aplikacija</p> |
| 1.4.1 | <p>Osnovni pregled</p> <p>Korisnik u interfejsu vidi listu aplikacija koje su trenutno postavljene na server.</p> |
| 1.4.2 | <p>Status aplikacije</p> <p>Korisnik u listi pored imena aplikacije ima prikazanu ikonicu koja oznacava da li je aplikacija pokrenuta ili ne.</p> |
| 1.4.3 | <p>Pokretanje aplikacije</p> <p>Korisnik u interfejsu za pregled ima pristup dugmetu za pokretanje aplikacije, a ako je aplikacija vec pokrenuta, dugmetu za gasenje aplikacije.</p> |
| 1.4.4 | <p>Azuriranje aplikacije</p> <p>Korisnik ima pristup dugmetu za azuriranje aplikacije ako aplikacija nije pokrenuta.</p> |
| 1.4.5 | <p>Pregled logova aplikacije</p> <p>Korisnik ima u bilo kom trenutku pristup pregeledu logova bilo kog koraka postavljanja aplikacije ili njenog pokretanja.</p> |

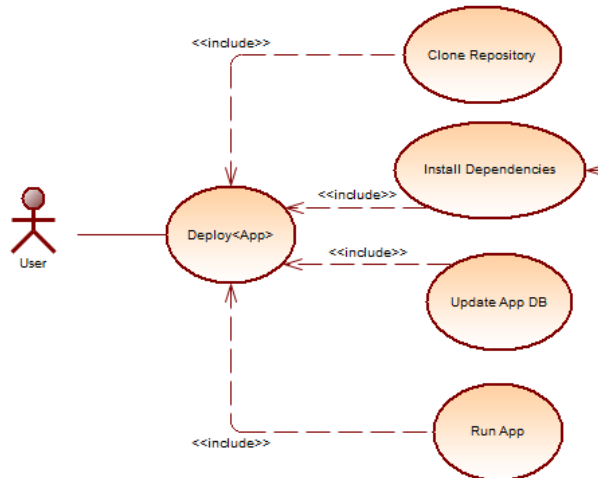
Tabela 1 Funkcionalni zahtevi korisnika

| Zahtev | Opis |
|--------|--|
| 2.1 | Bezbednost |
| 2.1.1 | Login Korisnicki interfejs mora biti obezbedjen login sistemom pomocu korisnickog imena i šifre |
| 2.1.2 | API API Goose sistema mora biti pristupacan samo ulogovanim korisnicima. Ogranicavanje pristupa se ostvaruje preko JWT tokena. |
| 2.2 | Logging |
| 2.2.1 | HTTP Sistem loguje sve HTTP zahteve u fajl kojem korisnik ima pristup. |
| 2.2.2 | Segmentisani logging Logovi HTTP zahteva se cuvaju u fajlovima za svaki pojedinačni dan. |
| 2.3 | Pristup serveru |
| 2.3.1 | SSH Server na kome je pokrenut Goose sistem treba da ima omogucen SSH servis radi pristupa zbog održavanja ili u slučaju kvara. |
| 2.4 | Karakteristike servera |
| 2.4.1 | Pristup internetu Fizicki server na kome je postavljen sistem mora imati pristup internetu. |
| 2.4.2 | Dependencies Server treba da ima instalirane runtime alate i kompajlere za Nodejs (v10.6.0 i novije), Python (v3.6 i novije), Go (v1.14 i novije) i Git. |
| 2.4.3 | Routing Ruter lokalne mreže treba da ima preusmeren port 80 ka masini na kojoj je postavljen Goose sistem. |

Tabela 2 Nefunkcionalni zahtevi

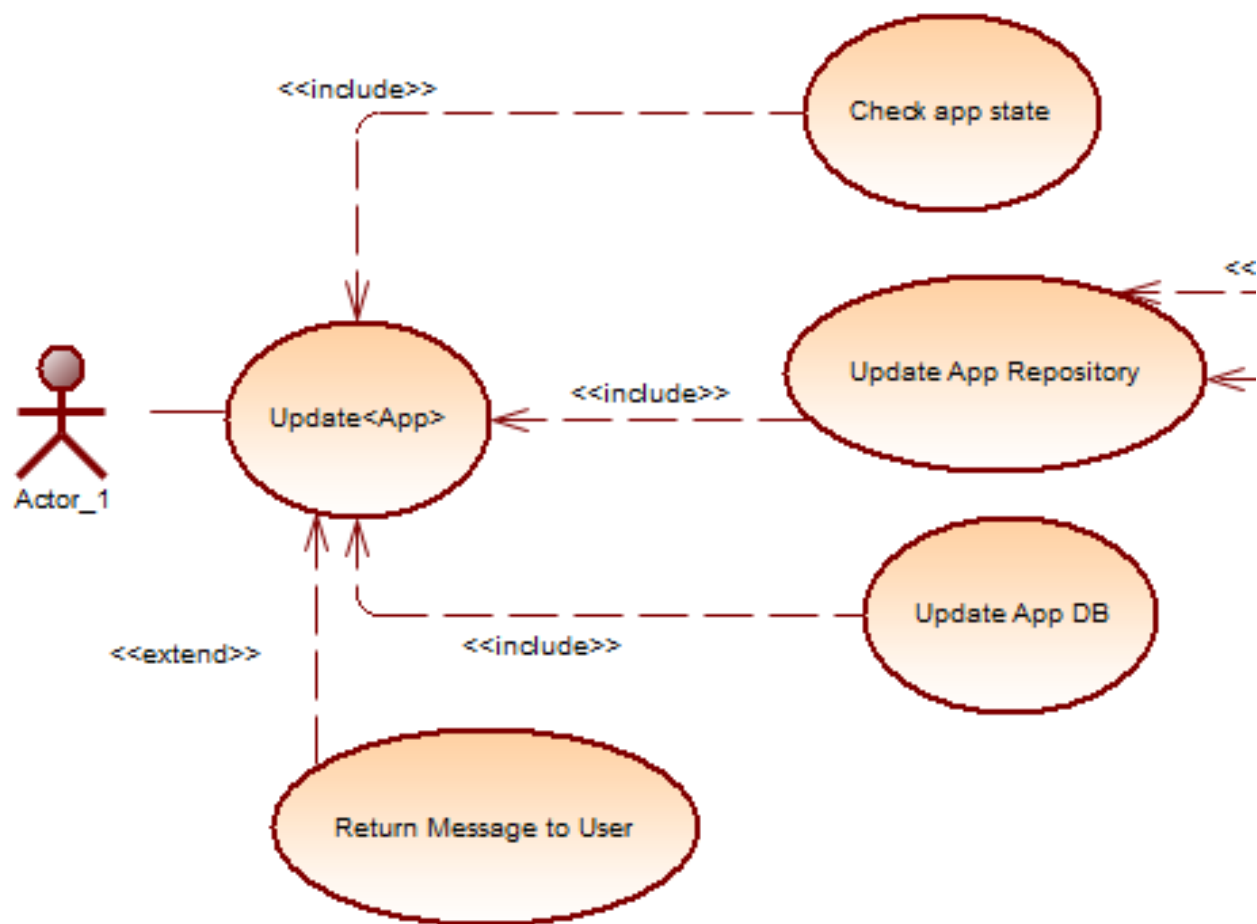
Slučajevi upotrebe

Slučaj upotrebe postavljanja aplikacije



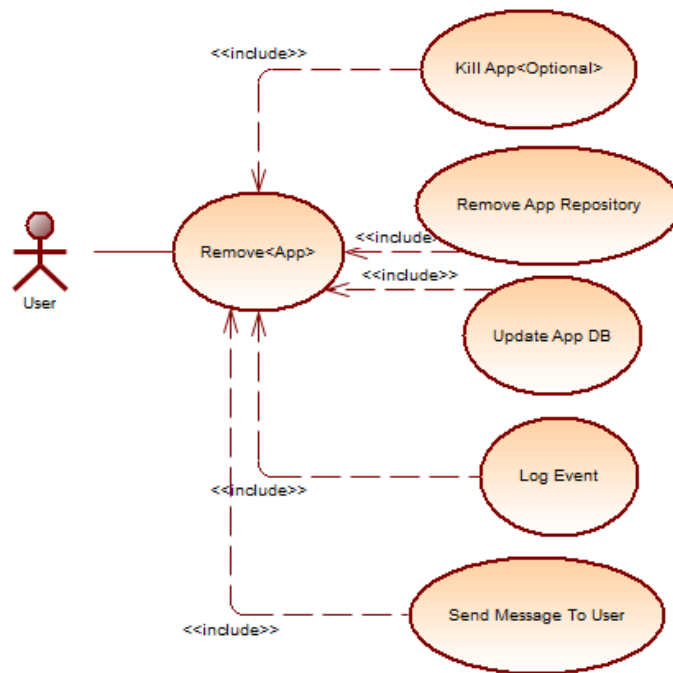
| | |
|------------------|---|
| Naziv | Postavljanje aplikacije na sistem |
| Aktor | Korisnik |
| Preduslovi | Nema |
| Okidač | Korisnik je izabrao opciju za postavljanje aplikacije |
| Normalni tok | <ol style="list-style-type: none">1. Korisniku se prikazuje za interfejs sa unosnim poljima za parametre za aplikaciju koju želi da postavi.2. Korisnik posle unošenja validnih podataka ima mogućnost da pošalje zahtev za postavljanje aplikacije.3. Korisnik potvrđuje unešene parametre i šalje zahtev za postavljanje.4. Korisniku se prikazuje informacija da je postavljanje aplikacije zapčeto [IZ - 1]5. Korisniku se prikazuje informacija da je aplikacija postavljena i aplikacija je dodata u listu postavljenih aplikacija. |
| Alternativni tok | Korisniku se prikazuje poruka o nevalidnim podacima koje treba da izmeni. |
| Izuzeci | [IZ - 1] Korisnik je uneo nevalidne podatke. Korisnik se vraća na interfejs za postavljanje aplikacije |
| Postuslovi | Korisnik je postavio aplikaciju |

Slučaj upotrebe ažuriranja aplikacije



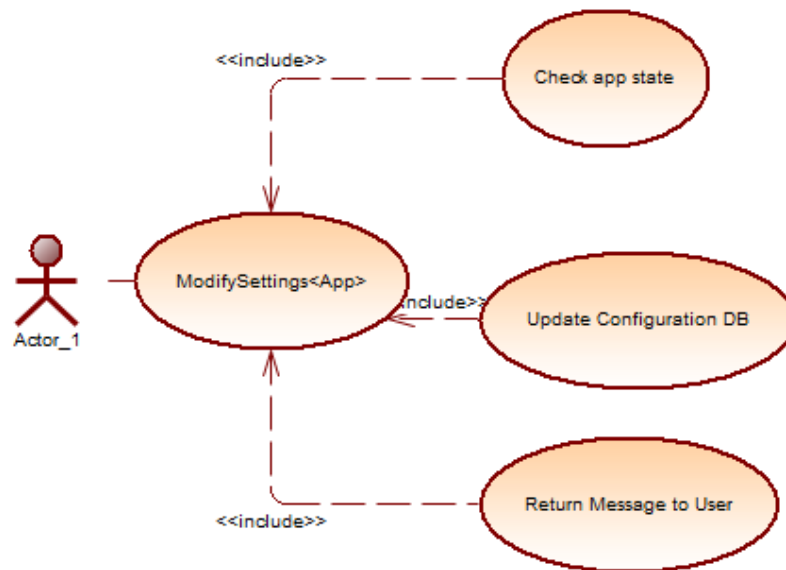
| | |
|------------------|--|
| Naziv | Ažuriranje postavljene aplikacije |
| Aktor | Korisnik |
| Preduslovi | Aplikacija nije aktivna |
| Okidač | Korisnik je izabrao opciju za ažuriranje |
| Normalni tok | 1. Korisniku se prikazuje informacija da je aplikacija ažurirana i u listi aplikacija se promenio datum poslednjeg ažuriranja. |
| Alternativni tok | |
| Izuzeci | |
| Postuslovi | Korisnik je ažurirao aplikaciju. |

Slučaj upotrebe brisanja aplikacije



| | |
|------------------|---|
| Naziv | Brisanje postavljene aplikacije |
| Aktor | Korisnik |
| Preduslovi | Aplikacija nije aktivna |
| Okidač | Korisnik je izabrao opciju za brisanje |
| Normalni tok | 1. Korisniku se prikazuje informacija da je aplikacija obrisana i aplikacija je uklonjena iz liste postavljenih aplikacija. |
| Alternativni tok | |
| Izuzeci | |
| Postuslovi | Korisnik je obrisao aplikaciju. |

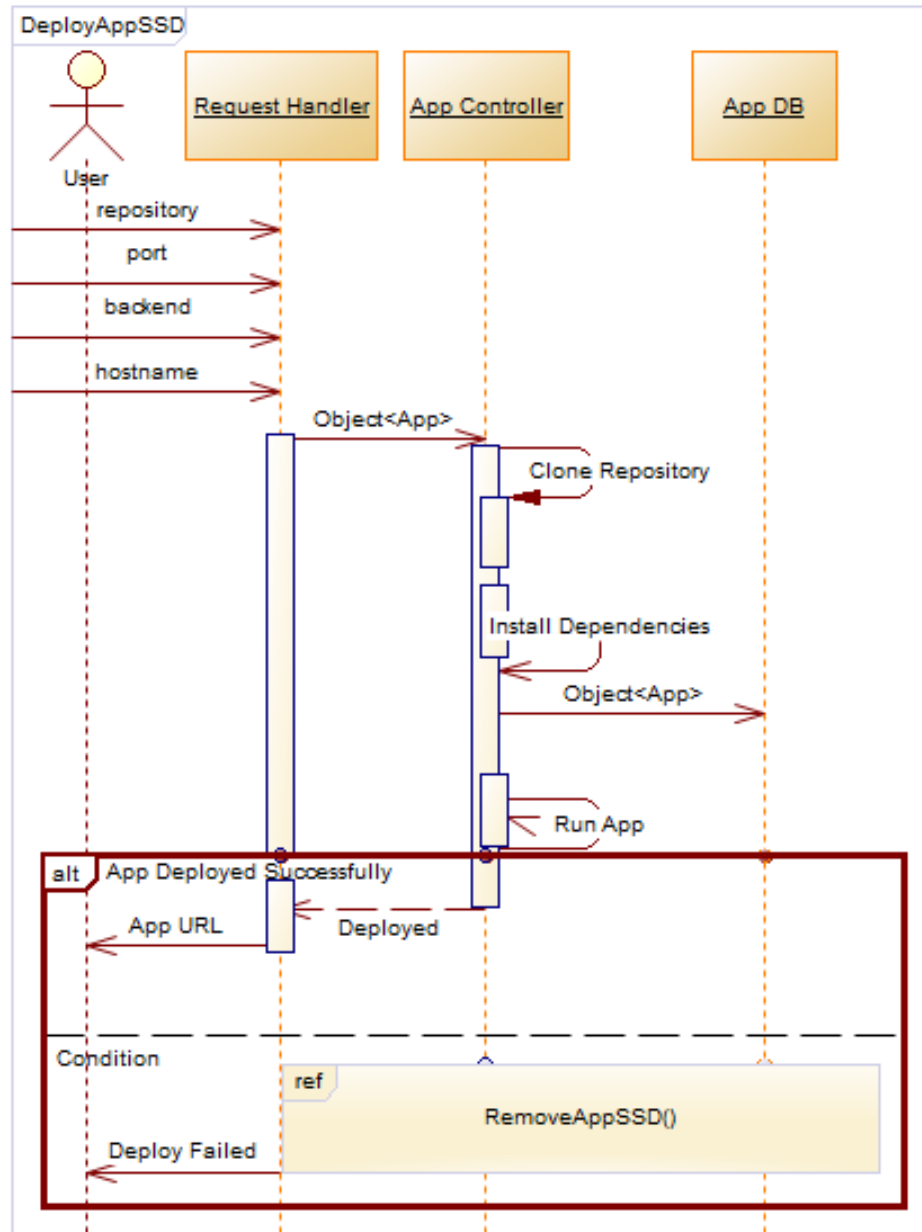
Slučaj upotrebe promene parametara aplikacije



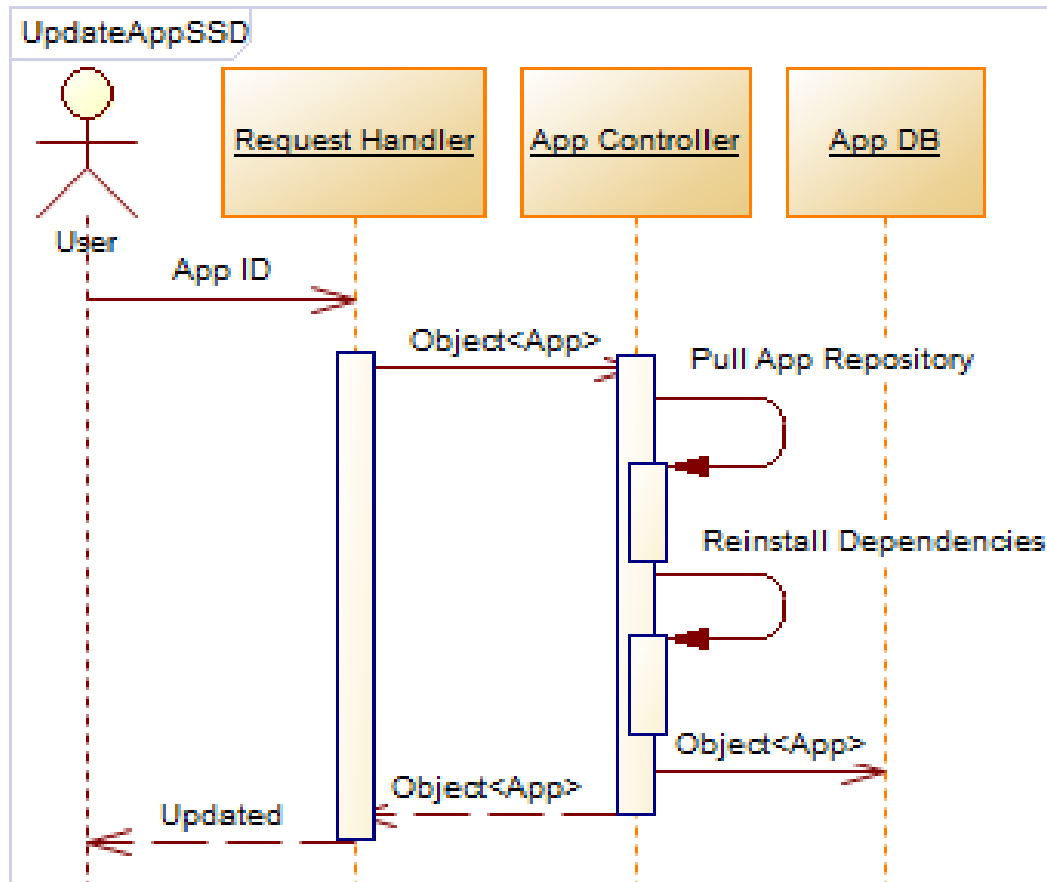
| | |
|------------------|---|
| Naziv | Izmena parametara postavljene aplikacije |
| Aktor | Korisnik |
| Preduslovi | Aplikacija nije aktivna |
| Okidač | Korisnik je izabrao opciju za izmenu parametara |
| Normalni tok | <ol style="list-style-type: none"> 1. Korisniku se prikazuje za interfejs sa unosnim poljima trenutnih parametara aplikacije. 2. Korisnik posle unošenja validnih podataka ima mogućnost da pošalje zahtev za promenu parametara. 3. Korisnik potvrđuje unešene parametre i šalje zahtev za promenu. 4. Korisniku se prikazuje informacija da su parametri promenjeni i novi parametri su azurirani u listi postavljenih aplikacija [IZ - 1]. |
| Alternativni tok | Korisniku se prikazuje poruka o nevalidnim podacima koje treba da izmeni. |
| Izuzeci | [IZ - 1] Korisnik je uneo nevalidne podatke. Korisnik se vraća na interfejs za promenu parametara. |
| Postuslovi | Korisnik je promenio parametre aplikacije. |

Dijagrami sekvenci

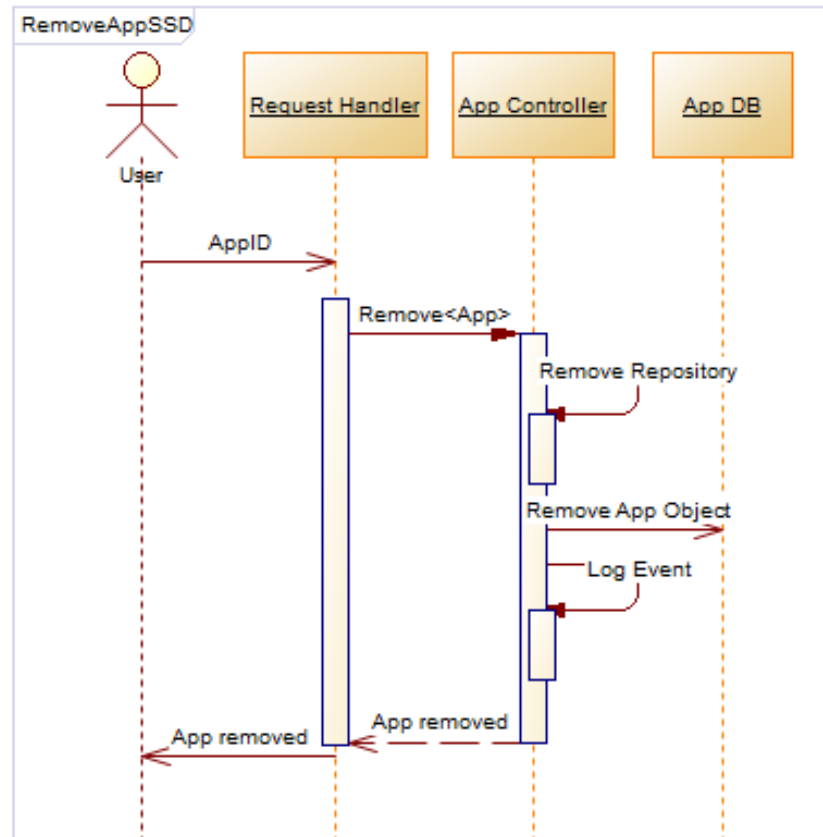
Dijagram postavljanja aplikacije



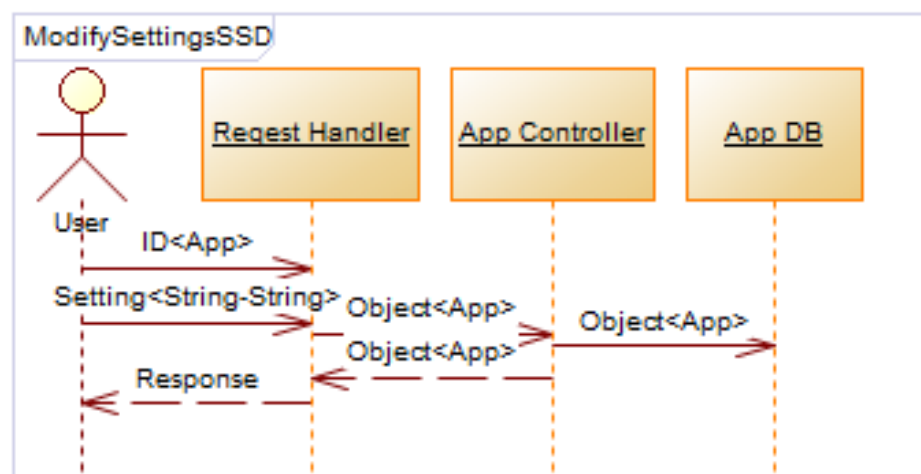
Dijagram ažuriranja aplikacije



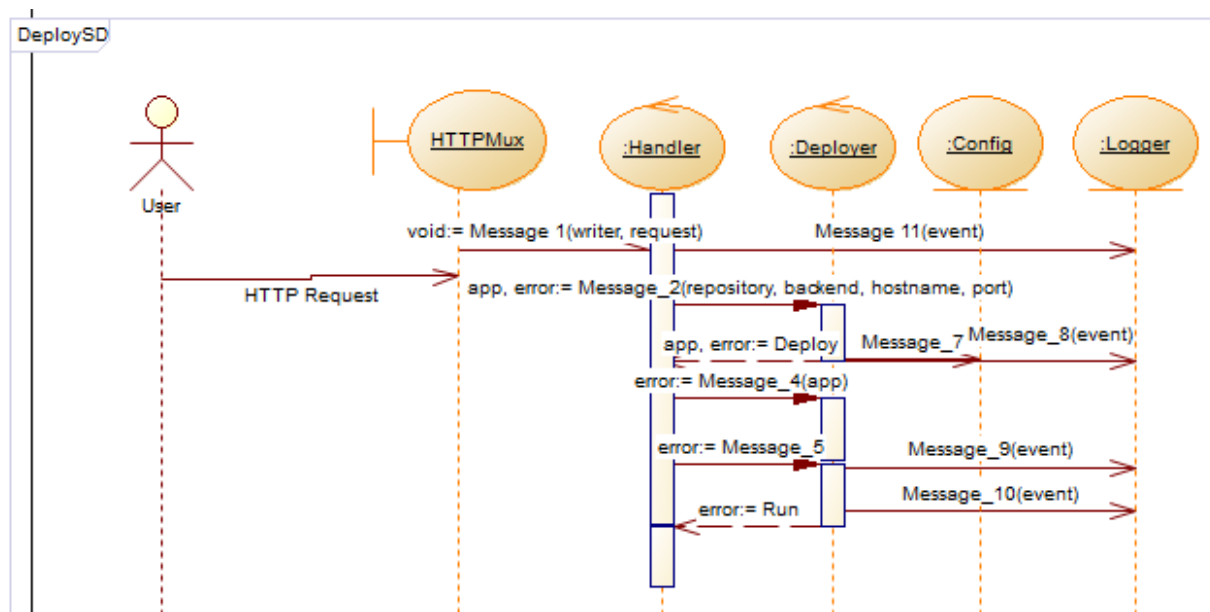
Dijagram brisanja aplikacije



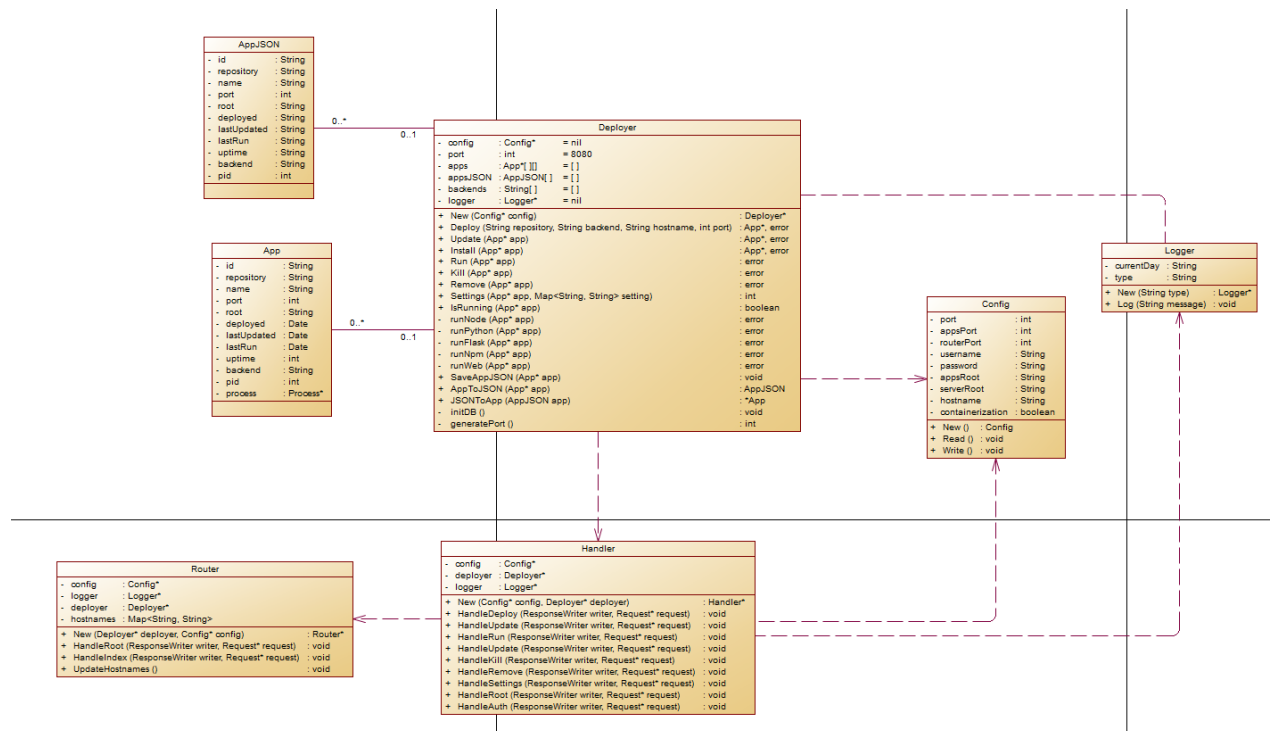
Dijagram promene parametara aplikacije



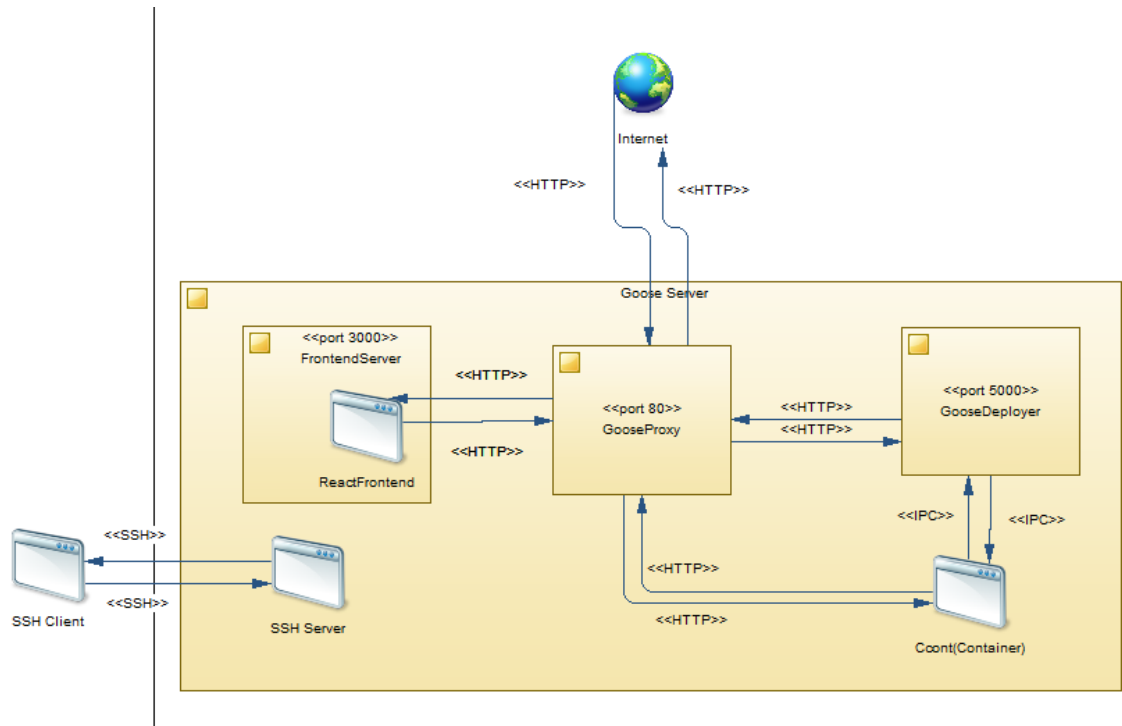
Deteljan dijagram postavljanja aplikacije



Klasni dijagram



Arhitektura sistema

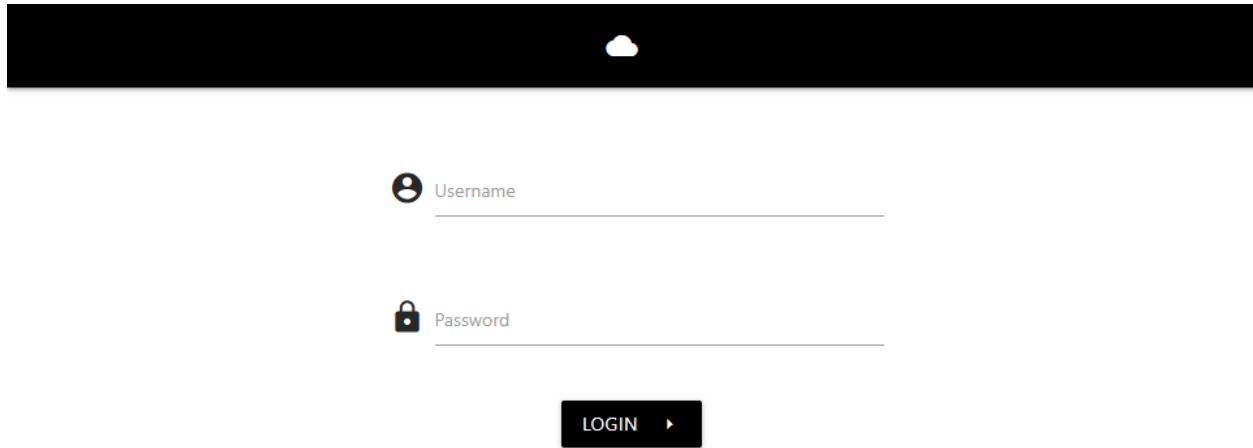


Opis arhitekture sistema

Za arhitekturu sistema odabran je repository patern arhitekture sistema gde se sve informacije cuvaju na jednom mestu a distribuirane su do korisnika preko servisno orijentisanog backenda koji sa klijentom komuncira preko HTTP-a. Cela arhitektura se nalazi iza proksi servera koji određuje da gde će se HTTP zahtev poslati u zavisnosti od hostname-a. Controller aplikacija (tj. njihovih kontejnera, Ccont) omogućava kontejnerima pristup internetu preko komunikacije sa proksijem. React frontend se takođe nalazi na odvojenom serveru i potpuno je autonoman. U suštini odžavamo ideju o modularnosti koju smo pomenuli ranije. Tako da na primer uklanjanjem frontenda ostaje web API koji moze koristiti bilo koji frontend interfejs ili pak neka druga aplikacija ili sistem.

Korisnički interfejs

Login forma





The login form features a dark header bar with a white cloud icon in the center. Below the header, there are two input fields: 'Username' with a person icon and 'Password' with a lock icon. A 'LOGIN' button with a right-pointing arrow is positioned below the password field.

Pregled postavljenih aplikacija



The overview section shows a dark header bar with a white cloud icon. Below the header, there is a circular button with a cloud icon and a plus sign. A table lists three deployed applications, each with a cloud icon and a checkmark:

| |
|---|
|  portfolio-react |
|  blog |
|  sh |

U listi vidimo postavljene aplikacije. Stiklir u oblaku pored imena aplikacije označava da je aplikacija pokrenuta. Dugme iznad liste je za pristup interfejsu za deployment.

Pregled informacija o pokrenutoj aplikaciji

portfolio-react

| | | | |
|-------|---|-----------|----------------------|
| ID: | Agf0unyZR | Deployed: | 21/01/2020, 04:45:05 |
| Name: | portfolio-react | Updated: | 21/01/2020, 04:45:05 |
| Repo: | http://github.com/7aske/portfolio-re... | Run: | 24/01/2020, 04:39:35 |
| Root: | /root/.local/share/goose/instances/p... | Uptime: | 1d 17h 32m 28s |
| Port: | 32815 | PID: | 14329 |
| Host: | http://portfolio.7aske.com | Backend: | |

Kill

blog

sh

U datom pregledu možemo videti sledece informacije o aplikaciji:

ID

Nasumično generisan jedinstveni identifikator pokrenute aplikacije.

Name

Naziv aplikacije izvučen iz imena repozitorijuma koji je korišćen.

Repo

URL do Github ili sličnog repozitorijuma koji je korišćen za deployment.

Root

Lokacija aplikacije na disku web servera

Port

Port na webserveru koji je rezervisan za aplikaciju

Host

URI hostname preko koga pristupamo aplikaciji

Deployed

Datum i vreme deploymenta

Updated

Datum i vreme poslednjeg ažuriranja aplikacije

Run

Datum i vreme poslednjeg pokretanja aplikacije

Uptime

Vreme rada aplikacije od poslednjeg pokretanja

PID

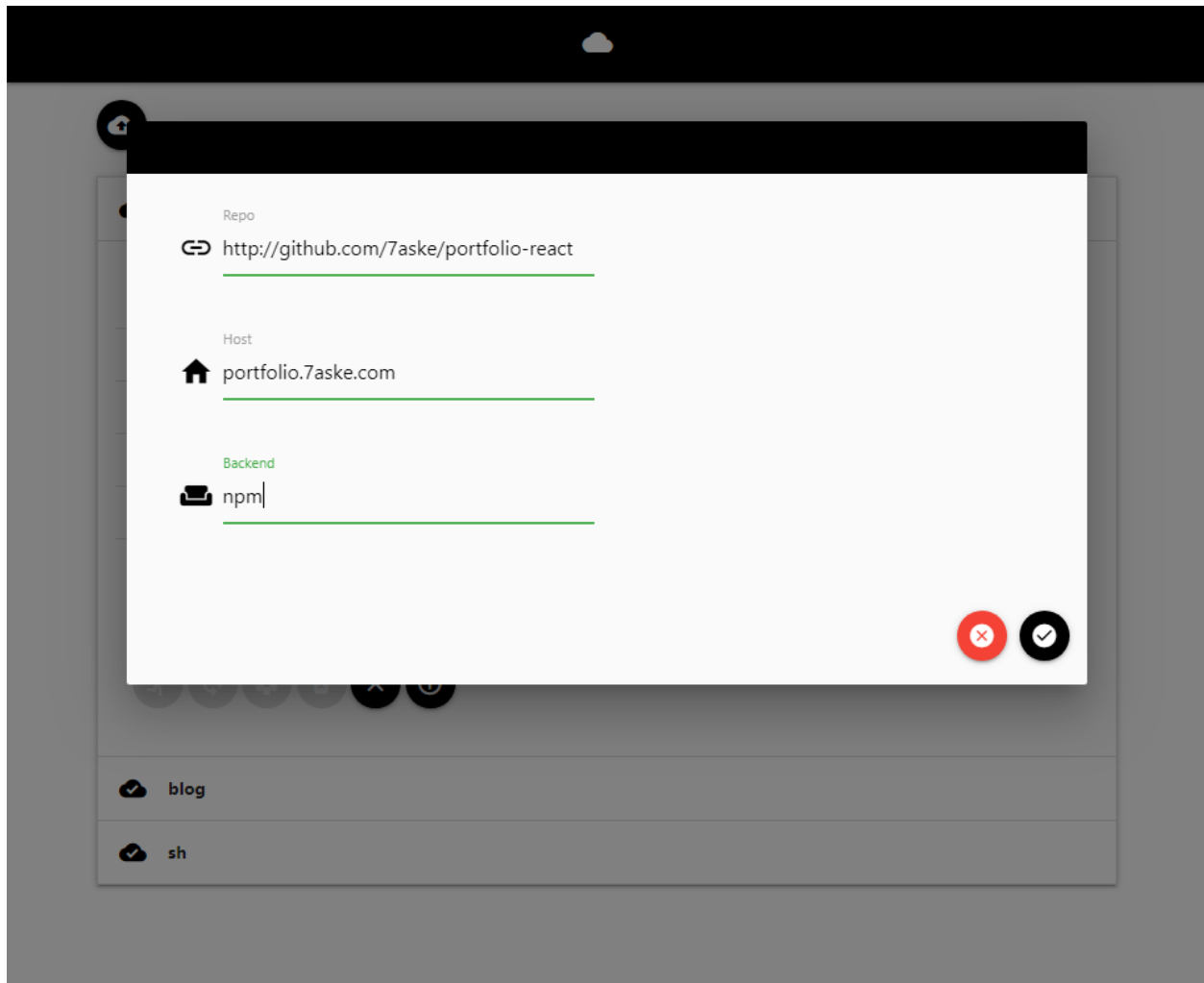
ID procesa na serveru koji je asociran aplikaciji

Backend

Backend sistem koja je korišćen za deployment

U dnu liste vidimo kontrole koje predstavljaju interfejse za, redom, pokretanja ažuriranje, menjanje podešavanja, brisanje, gašenje i čitanje logova. Dugmeta za logout nema jer se svi korisnički podaci čuvaju u sesiji trenutno otvorenog tab-a tako da čim je tab zatvoren korisnik je automatski izlogovan. Interfejs se osvežava svakih 10 sekundi da bi korisnik imao koliko toliko realne informacije o status svojih aplikacija.

Deployment interfejs



The image shows a deployment interface with a modal form. The form has three sections: 'Repo' with a GitHub icon and the URL 'http://github.com/7aske/portfolio-react'; 'Host' with a house icon and the domain 'portfolio.7aske.com'; and 'Backend' with a server icon and the text 'npm'. Each section has a green underline. At the bottom right of the form are two circular buttons: a red one with a white 'x' and a black one with a white checkmark. Below the form, there is a list of items: 'blog' and 'sh', each preceded by a checkmark icon.

Na ovom interfejsu vidimo formu za postavljanje aplikacije na sistem.

Repo

URL do Github ili sličnog repozitorijuma koji je korišćen za deployment.

Host

URI hostname preko koga pristupamo aplikaciji

Backend

Bekend sistem koja koristimo za deployment. Trenutno podržani sistemi su Node, Npm, Python i Flask.

Testiranje

Testiranje modula proverava da li pojedinačne funkcije i komponente ispravno funkcionišu sa svim očekivanim tipovima ulaza, u skladu sa dizajnom komponente. Jedinično testiranje treba vršiti u kontrolisanom okruženju gde rezultati metoda mogu biti precizno proučen i biti im određena validnost.

Go programski jezik u sebi ima ugrađeno jedinično testiranje koje se veoma lako pokreće komandom `go test` i koja testira funkcije na nivou paketa. U ovom slučaju testiramo da li će nam program, preciznije funkcija za instalaciju izbaciti grešku ako koristimo nevalidan backend koji nije podržan.

```
package deployer

import (
    "../internal/deployer"
    "../internal/instance"
    "testing"
)

func TestInstall(t *testing.T) {
    bkend := "golang" // invalid backend
    inst := instance.New(
        "https://github.com/7aske/goose",
        "goose.7aske.com",
        instance.Backend(bkend))
    json := instance.ToJSONStruct(inst)
    err := deployer.Deployer.Install(json)
    if err == nil {
        t.Errorf("Backend check failed, invalid backend %s was installed", bkend)
    }
}
```

U našem slučaju test prolazi i baca grešku 'backend not implemented', što je i očekivano.

PASS

ok _/home/nik/Code/go/goose/test 0.004s

Zaključak

Obzirom na to da je sistem već bio dizajniran pre nastanka ove dokumentacije a ponovnom revizijom usked analize za izradu iste došlo do potpunog re-dizajna koji je za rezultat imao mnogo bolji sistem primećujem da sva znanja dobijena na ovom predmetu i tokom izrade ove dokumentacije će doprineti mnogim drugim budućim projektima.