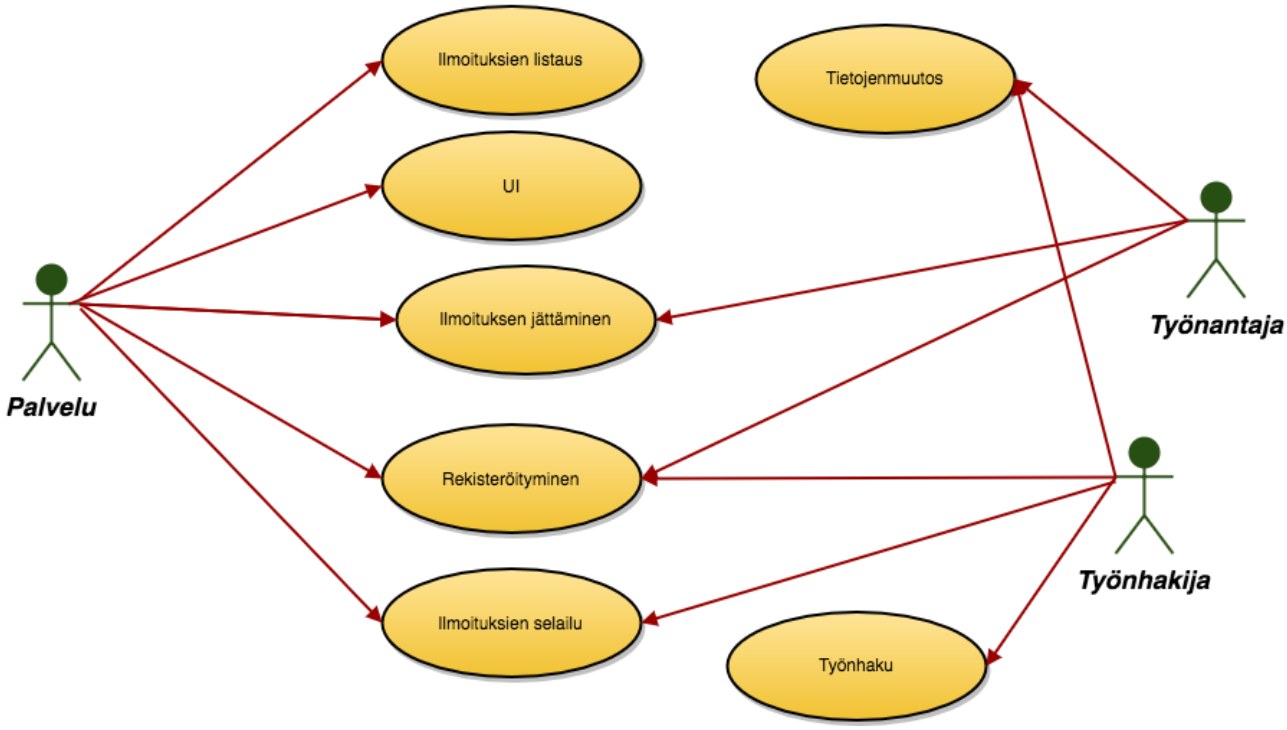
1. Johdanto

Järjestelmä on tarkoitettu helpottamaan työnhakua. Työnhakija voi selata työpaikka ilmoituksia ja vastaa sähköpostilla ilmoituksen jättäjälle. Työnantajat voivat julkaista työpaikka ilmoituksia ilmaiseksi.
Sivusto toimii verkossa sille asetetun IP osoitteen alla, sen näkymät on toteutettu HTML ja CSS kielillä, missä tyylien määrittämiseen on käytetty apuna Bootstrap ja Kickstart kirjastoja, HTML sivujen moottorina toimii Twig php pohjainen sivupohja moottori. Palvelin puolen ohjelmointi on toteutettu PHP kielellä käyttäen Slim micro kirjastoa, ja ohjelman sisäinen malli noudattaa MVC mallinnusta.
Tietokantana toimii ensisijaisesti PostgreSQL pohjaista tietokantaa, mutta dev-ops ympäristöön on myös jätetty tuki MySQL tietokannalle. Projektissa on hyödynnetty valmista Tsoha-Bootstrap boiler plate projektia githubista.

Ensisijaisesti palvelimena toimii users.cs.helsinki.fi palvelin, mutta vaihtoehdoiksi on myös määritelty Digitalocean droplet ja Hostinger.

2. Yleiskuvaus



- Palvelu:
  Palvelulla tarkoitetaan www palvelua jossa palvelu näkyy ja missä työnhakijat ja työnantajat voivat luoda työpaikkoja sekä etsiä työpaikkoja.
- Työnhakija
  Työnhakijalla tarkoitetaan käyttäjää joka käyttää palvelua työnhakuun eli selailemaan ilmoituksia, rekisteröitymään ja hakemaan työpaikkoja.
- Työnantaja

Työnantajalla tarkoitetaan työntarjoajaa joka rekisteröityy ja ilmoittaa työpaikan.
Palvelu käyttötapaukset:

- Ilmoituksien listaus:
  Kuka tahansa voi lukea ilmoituksia, hakea ilmoituksia, ja selata sivuja, mutta hän ei voi hake työpaikkoja.
- UI:
  UI:lla tarkoitetaan sivujen ulkoasua joka on suunniteltu käyttäjän käyttökokemuksen maksimoimiseksi.
- Ilmoituksen jättäminen:
  Rekisteröitynyt työnantaja voi jättää työpaikka ilmoituksen omien sivujensa kautta.
- Rekisteröityminen:
  Kuka tahansa voi rekisteröityä palveluun joko työnantajana tai työnhakijana.

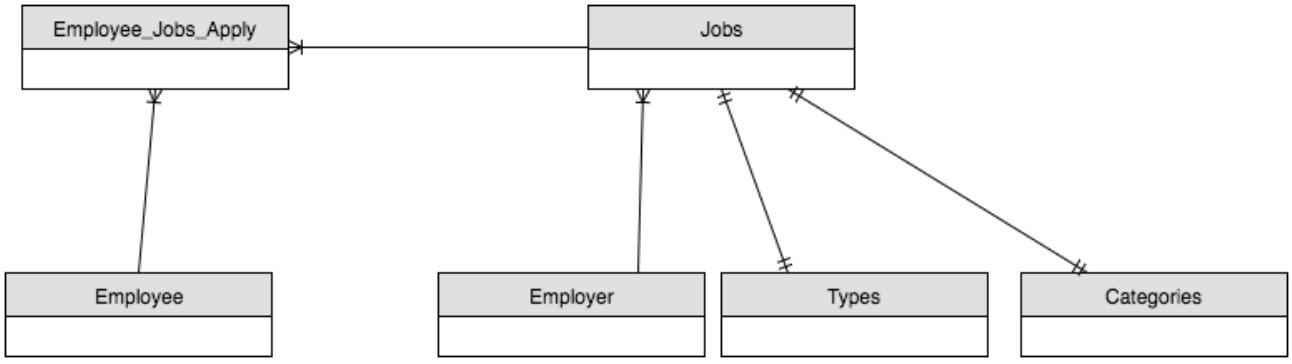
Työnhakijan käyttötapaukset:

- Työnhakija voi rekisteröityä palveluun jonka jälkeen hän voi selata ja hakea työpaikkoja sekä muuttaa tietojaan.

Työnantajan käyttötapaukset:

- Rekisteröityminen:
  Työnantaja voi rekisteröityä palveluun.
- Tietojen muutos:
  Työnantaja voi rekisteröitymisen jälkeen muuttaa tietojaan.
- Ilmoituksen jättäminen:
  Työnantaja voi jättää ilmoituksen uudesta työpaikasta.

3. Järjestelmän tietosisältö



[Kuvaus on muotoa Attribuutti, Arvojoukko, Kuvailu.]

Tietokohde: Employee
Id, Primary key, Employee taulun tietueen yksilöivä tunnus
First_name, merkkijono max 50 merkkiä, employee etunimi
Last_name, merkkijono max 100 merkkiä, employee sukunimi
Email, merkkijono max 100 merkkiä, employee sähköposti osoite
Username, merkkijono max 50 merkkiä, employee käyttäjätunnus

Password, merkkijono max 50 merkkiä, employee salasana
Description, teksti kenttä, employee esittely
Created, päiväys, employee luonti päiväys

Employee voi olla monta haettua Jobs, ja yhdellä Jobs monta employee hakijaa. Many to Many suhteen takia väliin on asetettu liitostaulu.

Tietokohde: Employer
Id, primary key, employer yksilöivä tunnus
Company_name, merkkijono max 50 merkkiä, employer yrityksen nimi
Email, merkkijono max 100 merkkiä, employer sähköposti
Username, merkkijono max 50 merkkiä, employer käyttäjätunnus
Password, merkkijono max 50 merkkiä, employer salasana
Company_description, teksti kenttä, employer esittely
Created, päiväys, employer luonti päiväys

Employer voi olla monta Jobs, mutta yhdellä työpaikalla voi olla vain yksi employer.

Tietokohde: Jobs
Id, primary key, jobs yksilöinti tunnus
Category_id, foreign key, categories taulun liitosavain
Employer_id, foreign key, employer taulun liitosavain
Type_id, foreign key, types taulun liitosavain
Description, teksti, job esittely teksti
Area, merkkijono max 25 merkkiä, job alue
Created, päiväys, job luonti päiväys

Jobs voi olla monta employee (hakijaa), ja yhdellä jobs on yksi employer. Yhdellä jobsilla on yksi types sekä yksi categories.

Tietokohde: Types
Id, primary key, types yksilöinti tunnus
Name, merkkijono max 50 merkkiä, types nimi
Color, merkkijono max 20 merkkiä, types väri joka renderoidaan rgba muodossa merkkijonon mukaan

Yhdella types voi olla yksi jobs, ja yhdellä jobs yksi types.

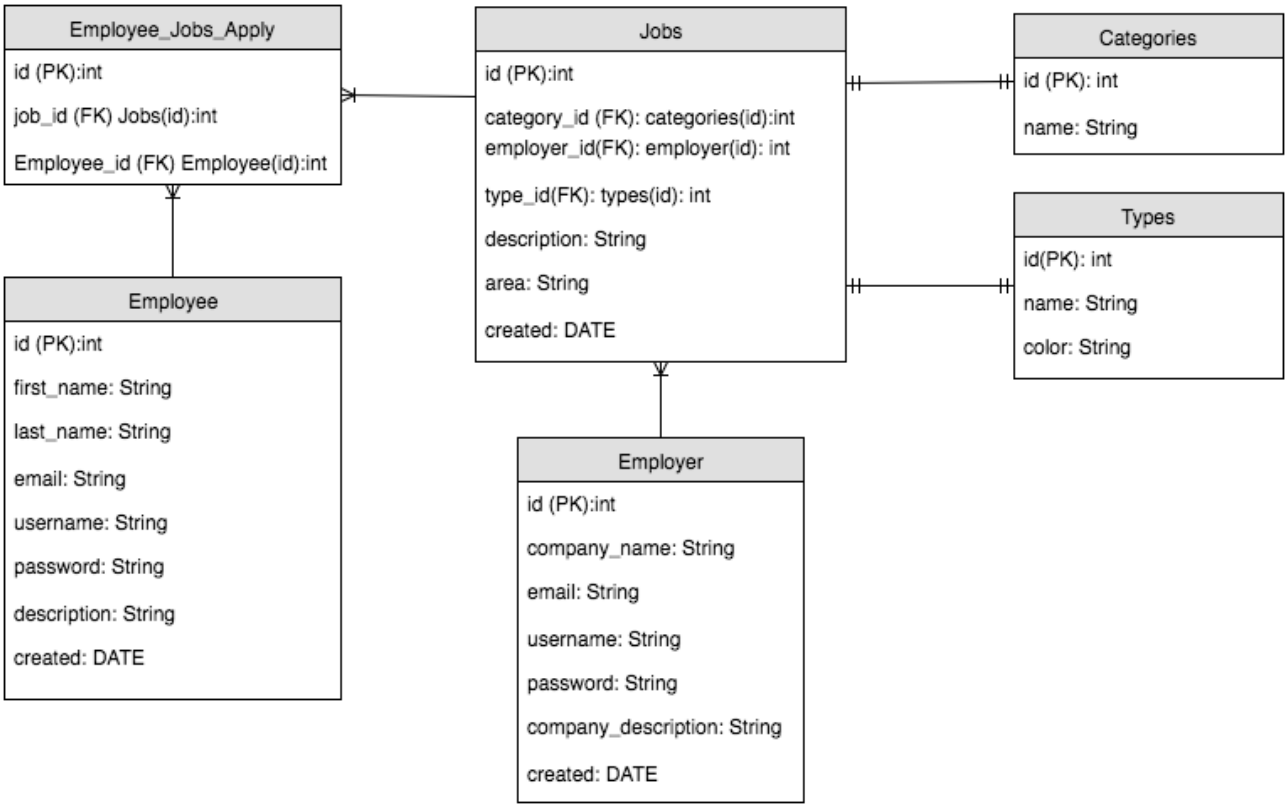
Tietokohde: Categories
Id, primary key, categories yksilöinti tunnus
Name, merkkijono max 100 merkkiä, categories nimi

Yhdellä categories voi olla yksi jobs, ja yhdellä jobs yksi categories.

Tietokohde: Liitostaulu→Employee_Jobs_Apply
Id, primary key, liitoksen yksilöinti tunnus
Job_id, foreign key, jobs taulun liitos avain

Employee_id, employee taulun liitos avain
Koska yhdellä jobs voi olla monta employee (hakijaa) ja yhdellä employee voi olla monta jobs niin Employee_Jobs_Apply toimii liitostauluna jossa yhdellä Employee voi olla monta Employee_Jobs_Apply ja Jobs monta Employee_Jobs_Apply.

4. Relaatiotietokantakaavio



5. Järjestelmän yleisrakenne

Sovelluksen toteutuksessa on noudatettu MVC mallia, joka jakaa sovelluksen kontrollereihin, malleihin ja näkymiin. Riippuvuuksien hallinnassa on käytetty Composer riippuvuuksien hallitsija ohjelmaa. Html sivujen sivumoottorina on käytetty Twig php sivupohjamoottoria. Sovelluksen polut on määritelty hakemistossa config/routes.php.

- Kontrollerit löytyvät hakemistosta app/controllers.
- Mallit löytyvät hakemistosta app/models
- Näkymät löytyvät hakemistosta app/views

Kontrollerit:

JobsController:

- JobsControllerin funktio *index* näyttää etusivun, eli hakee kaikki palveluun lisätyt duunit ja lisää ne mallille esitettäväksi.
- Funktio *addjob* esittää employer sessiossa näytettävää addjob nappia joka ohjaa addjob.html sivulle jos session palautta true.
- Funktio *addjob_handler* ottaa vastaan addjob lomakkeen tiedot ja luo niistä uuden jobs olion joka tallennetaan tietokantaan.
- Funktio *details* renderoi parametrinä saadun id:n mukaisen jobsin ja asettaa näkymään tarvittavat muuttujat.
- *deletejob* funktio poistaa *id*:n mukaisen jobin palvelusta ja tietokannasta.
- *apply* funktio näkyy employee sessiossa ja se luo uuden EmployeeJobsApply olion joka tallenetaan tietokantaan. Kyseinen olio toimii liitostauluna taulujen Jobs ja employee välillä.

EmployerController:

- Funktio *login* renderoi employer_login näkymän.
- funktio *handle_login* suorittaa login lomakkeeseen syötettyjen tietojen vastaanoton sekä validoinnin Employer luokan *auth* funktion kautta ja asettaa Session sekä ohjaa employer omille sivuille jos true.
- funktio *register* luo employerin rekisteröinti näkymän.
- funktio *handle_register* ottaa vastaan lomakkeen tiedot, validoi ne ja luo uuden employer olion jos validointi onnistuu, sekä tallentaa olion tietokantaan. Jos true, niin luodaan uusi sessio oliolle ja ohjataan omille sivuille, muuten näytetään virhe ilmoitus.
- *Employe*r funktio esittää parametrinä saadun id:n mukaisen employer olion omat sivut.
- Funktio *logout* tappaa session ja kirjaa employer ulos.
- *update* funktio päivittää tietokantaan ja palveluun id:n mukaisen employer olion tiedot.
- *delete* funktio poistaa tietokannasta id:n mukaisen employer olion.

EmployeeController:

- Funktio *login renderoi* employee_login näkymän.
- funktio *handle_login* suorittaa login lomakkeeseen syötettyjen tietojen vastaanoton sekä validoinnin *Employe*e luokan auth funktion kautta ja asettaa Session sekä ohjaa employee omille sivuille jos true.
- funktio *register* luo employeein rekisteröinti näkymän.
- funktio *handle_register* ottaa vastaan lomakkeen tiedot, validoi ne ja luo uuden employee olion jos validointi onnistuu, sekä tallentaa olion tietokantaan. Jos true, niin luodaan uusi sessio oliolle ja ohjataan omille sivuille, muuten näytetään virhe ilmoitus.
- *Employee* funktio esittää parametrinä saadun id:n mukaisen employee olion omat sivut.
- Funktio *logout* tappaa session ja kirjaa employee ulos.
- *update* funktio päivittää tietokantaan ja palveluun id:n mukaisen employee olion tiedot.
- *delete* funktio poistaa tietokannasta id:n mukaisen employee olion.

Employer ja Employee-Controllers ovat oikeastaan samat koska en lähtenyt aikataulunpuitteissa vielä toteuttamaan isompia erinnäisiä toimintoja joita alkuun suunnittelin.

HelloWorldController:

- Malli kontrolleri. Ei tee mitään. Testaukseen ja debuggaukseen tarkoitettu.

Mallit:

Categories:

- funktio *all* palauttaa kaikki Categories taulun tietueet.
- funktio *findOne($name)* palauttaa yhden categories taulun tietueen id:n (avaimen) name parametrin perusteella.

Employee:

- Funktio *validate* saa parametreina employee olion parametrit, ja hyödyntää Valitron kirjaston tarjoamaa palvelua parametrien validointiin.
- *allEmployees* funktio palauttaa kaikki Employees taulun tietueet taulukkona.
- Funktio *newEmployee* luo uuden Employee olion ja tallentaa sen tietokantaan.
- *find* funktio hakee yhden Employee olion tietokannasta id: parametrin perusteella.
- *updateEmployee* funktio päivittää Employee olion tiedot tietokantaan.
- *deleteEmployee* funktio poistaa Employee olion tietokannasta.
- Funktio *auth* saa parametreinä käyttäjätunnuksen ja salasanan, ja hakee parametrien perusteella tietokannasta Employee olion, jos funktio palauttaa true niin olio löytyi ja authentikaatio voidaan suorittaa, jos taas false niin oliota ei löytynyt.
- Funktio *checkUsername* tarkistaa onko parametrinä annettu käyttäjätunnus jo käytössä.

EmployeeJobsApply:

- Funktio *apply* luo uuden Employee_Jobs_Apply olion job_id ja employee_id ulkoisilla avaimilla.
- Funktio *all* palauttaa kaikki tietueet taulukkona.
- Toimii liitoksena Jobs ja Employee välillä.

Employer:

- Funktio *validate* saa parametreina employer olion parametrit, ja hyödyntää Valitron kirjaston tarjoamaa palvelua parametrien validointiin.
- *allEmployers* funktio palauttaa kaikki Employers taulun tietueet taulukkona.
- Funktio *newEmployer* luo uuden Employer olion ja tallentaa sen tietokantaan.
- *find* funktio hakee yhden Employer olion tietokannasta id: parametrin perusteella.
- *updateEmployer* funktio päivittää Employer olion tiedot tietokantaan.
- *deleteEmployer* funktio poistaa Employer olion tietokannasta.

- Funktio *auth* saa parametreinä käyttäjätunnuksen ja salasanan, ja hakee parametrien perusteella tietokannasta Employee olion, jos funktio palauttaa true niin olio löytyi ja authentikaatio voidaan suorittaa, jos taas false niin oliota ei löytynyt.
- Funktio *checkUsername* tarkistaa onko parametrinä annettu käyttäjätunnus jo käytössä.

Jobs:

- Funktio *all* palauttaa kaikki Jobs taulukon tietueet taulukkona.
- Funktio *find* palauttaa id parametriä vastaavan yhden Jobs tietueen.
- Funktio *findOne* palauttaa id parametriä vastaavan Jobs tietueen id avaimen (oikeastaan turha funktio mutta helpotti tietyn lisäyksen toteuttamista)
- *FindByParam* funktio on tarkoitettu hakuja varten mutta jäi toteuttamatta tässä vaiheessa.
- *findByEmploye*r funktio on myös toteuttamton funktio yhden jobsin palauttamiseen employer.id:n mukaan.
- *newjob* funktio suorittaa uuden Jobsin lisäämisen tietokantaan.
- *deleteJob* funktio poistaa Jobs olion tietokannasta.

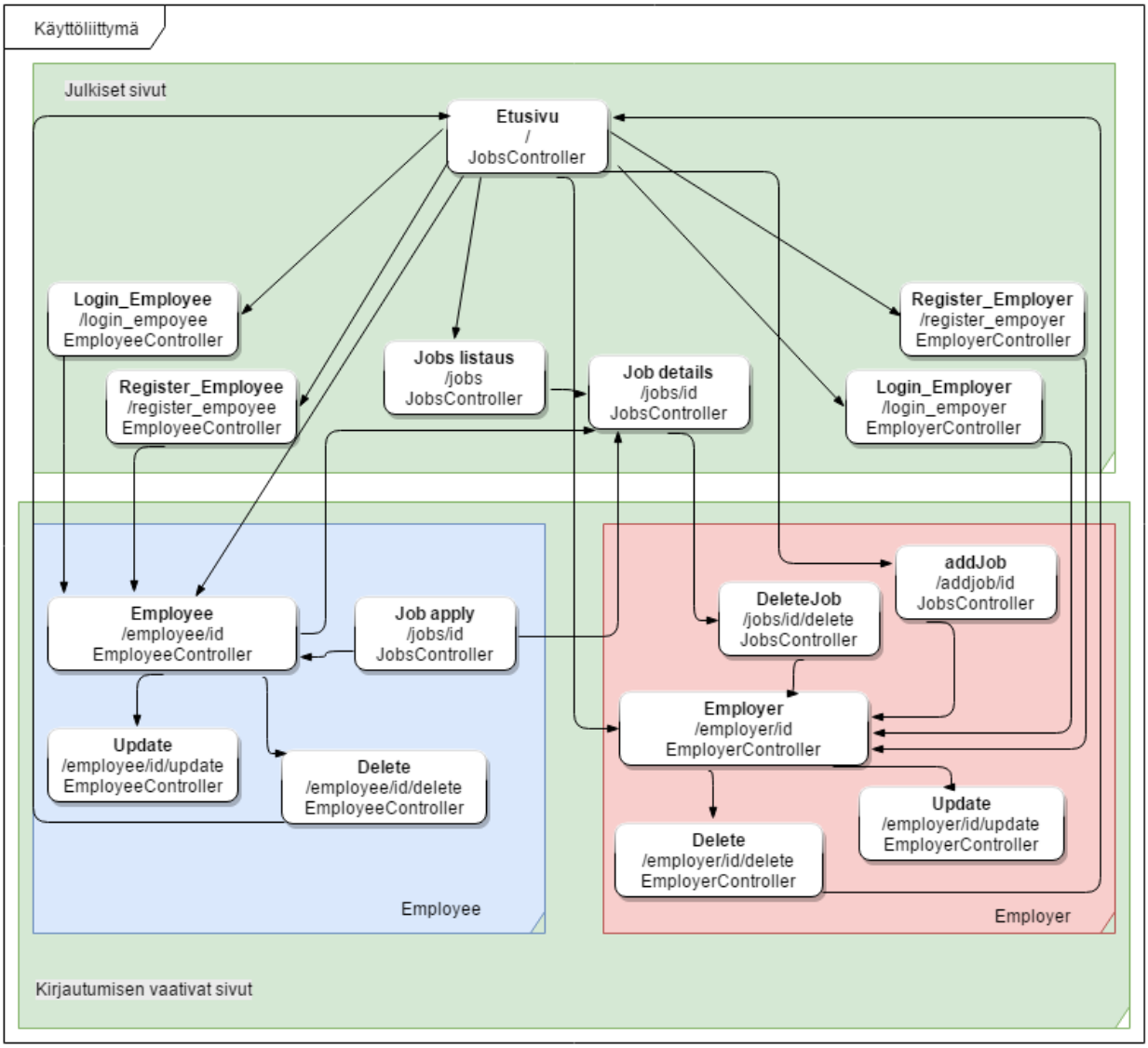
Types:

- funktio *allTypes* palauttaa kaikki Typess taulun tietueet.
- funktio *findOne($name)* palauttaa yhden Types taulun tietueen id:n (avaimen) name parametrin perusteella.

Views:

Views/employee sisältää employee kohtaiset html näkymät. Views/employer sisältää employer kohtaiset html näkymät. Views/jobs sisältää jobs kohtaiset html näkymät ja views/suunnitelmat sisältää plain html suunnitelmat näkymiä varten.

6. Käyttöliittymä ja järjestelmän komponentit



7. Asennustiedot

Sovellus on asennettuna osoitteeseen http://mikvalla.users.cs.helsinki.fi/tsoha/
Käyttääksesi sovellusta luo itsellesi Duunin tarjoaja sekä Duunin hakija tunnukset.

8. Käynnistys - /Käyttöohje

Helsingin Yliopiston users-palvelin

Helsingin Yliopiston users-palvelimelle sovelluksen voi pystyttää helposti. Ota yhteys usersiin komentoriviltä ssh kautta, komenna *wanna-htdocs* sekä *wanna-postgres* jos et ole vielä niin tehnyt. Sitten lisää tedostoon *config/environment.*sh omat asetuksesi.

Seuraavaksi mene komentorivillä projektisi juureen (esim. Tsoha-bootstrap) ja anna seuraavat komennot tässä järjestyksessä:

1. Bash deploy.sh
2. Bash create_tables.sh
3. Bash add_test_data.sh

Sovellus on nyt valmiina käyttöösi. Polussa */tietokantayhteys* näät tietokanta taulusi sekä testi datan.

Muut palvelimet

Yllä mainituilla askelilla pystyt asentamaan sovelluksen myös muulle palvelimelle olettaen että siihen on jo asennettu php-tuki, mutta tämä vaatii komento skriptien muuttamista (.sh tiedostot). Jos palvelimella on postgresql tuki niin voit käyttää valmiita create_table.sql ja add_test_data.sql komentoja. Jos kuitenkin palvelin tukee vain MySql:ää niin mene phpMyAdmin ohjelmaan palvelimella ja asenna tietokanta manuaalisesti MySQL notaatiota käyttäen. Tämän jälkeen joudut päivittämään sovelluksessa Models kansiosta löytyvien luokkien tietokanta haut vastaamaan MySql muotoilua.

NoSql tietokantoja käyttäessä suosittelen luomaan REST apin sovellukseen tai muuttamaan tietokantahakujen palautus muodon JSON muotoon. Tästä voi lukea lisää php:n dokumentaatiosta.

9. Testaus, tunnetut bugit ja puutteet & jatkokehitysideat

Aikarajoitteen ja perheellisten syiden takia sovellukseen ei tullut kaikkia toimintoja mitä olin alun perin suunnitellut. Puuttumaan jäivät:

- Jquery:lla suoritettu dynaamisuus
- Virhe viestien tarkka lajittelu ja näyttäminen
- Osa tyyleistä
- Hakutoiminto
- Paginaatio
- Details sivun hienompi tyylittely
- Employers logo lisäys
- Employee kuva + linkit ulkoisiin palveluihin
- OAuth2 kirjautuminen

- Käyttäjien erilliset oikeudet
- SQL kyselyt ovat todella geneerisiä, kyselyt pitäisi täsmentää.
- Area olisi parempi tietokanta tauluna

Jos ja kun kehitän toiminnot loppuun, on sovellus toimiva, moderni ja sitä voi käyttää julkisesti, esimerkiksi nerot.fi kilpailijana.