

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Oleg Barinov 120724

Development of customisable onboarding solution for remote-work environment

Bachelor's thesis

Supervisor: Sadok Ben Yahia
PhD
Professor

Tallinn 2021

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Oleg Barinov 120724

Kohandatava sisseelamisprogrammi lahenduse arendus kaugtöö keskkonna jaoks

Bakalaureusetöö

Juhendaja: Sadok Ben Yahia
PhD
Professor

Tallinn 2021

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Oleg Barinov

18.05.2021

Abstract

The goal of this thesis is to develop an onboarding platform for mid-sized companies.

In current pandemic times multiple organizations switched to the remote work environments. This had a negative influence on the retention rate of newly hired employees, also directly impacting the cost per hire rate. Based on conducted research two main problems of currently available onboarding solutions were identified: integrate ability and customization options.

In result, the main goal was achieved with both problems having a solution. However, level of customization options should be increased in order to fulfil requirements of bigger customers on the market.

Future development list was created which needs to be followed in order to become a competitive solution on a market.

This thesis is written in English and is 39 pages long, including 6 chapters, 15 figures.

Annotatsioon

KOHANDATAVA SISSEELAMISPROGRAMMI

LAHENDUSE ARENDUS KAUGTÖÖ KESKKONNA

JAOKS

Selle lõputöö eesmärk on arendada sisseelamisprogrammi platvorm keskmise suurusega firmadele.

Praeguse pandeemia ajal võtavad paljud organisatsioonid kasutusele kaugtöö keskkonna. Sellel oli negatiivne mõju värskelt palgatud töötajate lahkumise määrale ning mõjutas samuti otseselt värbamise kulu määra. Läbi viidud uuringu põhjal tuvastati kaks peamist probleemi hetkel olemasolevatest sisseelamisprogrammi lahendustest: integreerimise võimekus ja kohandamise valikud.

Lõpuks on põhieesmärk saavutatud mõlema probleemi lahendamiseks: värbajatel tekkis võimalus kohandatava sisu loomiseks ning samuti olid arendatud mõned veebiteenused. Kuigi, nagu teostatud uuring näitas, suuremate klientide vajaduste rahuldamiseks turul tuleks kohandamisvõimalusi siiski laiendada, kuna need on palju olulisemad kui teised lahenduse osad. Seetõttu sobib see lahendus rohkem väikeettevõtetele ja keskmise suurusega ettevõtetele.

Saadud andmete põhjal oli koostatud järgmiste sammude plaan. Selliste ideede teostamine on lihtne ja aitaks rakendust turule kiiremini viia, et saada konkurentsivõimeline lahendus. See pakuks ka huvi suurematele ettevõtetele.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 39 leheküljel, 6 peatükki, 15 joonist.

List of abbreviations and terms

DPI	Dots per inch
API	Application Programming Interface
REST	Representational State Transfer
UI	User Interface
AWS	Amazon Web Service
EC2	Elastic Compute Cloud
DNS	Domain Name System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JRE	Java Runtime Environment
ID	Identifier
JDK	Java Development Kit
CRUD	Create, read, update, delete
EU	European Union
NPM	Node Package Manager
GB	Gigabytes
DB	Database
SMTP	Simple Mail Transfer Protocol
IPv4	Internet Protocol version 4
JAR	Java ARchive
IP	Internet Protocol
YAML	Yet Another Markup Language
JSON	JavaScript Object Notation
JWT	JSON Web Token
SSL	Secure Sockets Layer
JS	JavaScript
SQL	Structured Query Language
WYSISYG	What You See Is What You Get

Template	Onboarding form shown to employee
NGINX	Web server
HR	Human Resources
IA	Department of Computer Systems

Table of contents

1 Introduction	11
1.1 Main task list	11
1.2 Problem statement and objectives	11
2 Technology overview	13
2.1 Overall architecture	13
2.2 PostgreSQL.....	14
2.3 Server instance.....	14
2.4 Back-end technologies.....	15
2.4.1 Lombok.....	15
2.4.2 JSON Web Token.....	15
2.4.3 Spring Security	15
2.4.4 Spring Boot.....	16
2.4.5 Hibernate and H2 in-memory database	16
2.5 Front-end technologies	16
2.5.1 Google maps widget	16
2.5.2 Quill text editor widget.....	17
2.5.3 Angular material.....	17
2.5.4 HTTP interceptor.....	17
2.5.5 Node Package Manager	17
2.5.6 Node.js.....	17
3 Server configuration	18
3.1 AWS instance creation	18
3.2 JDK and JRE installation.....	18
3.3 Virtual memory.....	18
3.4 Define back end as a service.....	18
3.4.1 Service configuration.....	19
3.4.2 YAML file properties	19
3.5 NGINX	20
3.6 Node.js and Yarn	21

3.7 DNS configuration.....	21
3.8 CERTBOT	21
3.9 PSQL installation and model	21
3.10 Code build and deployment	22
4 User interface and REST API.....	23
4.1 User Interface	23
4.1.1 Locations	23
4.1.2 Teams	24
4.1.3 Users	25
4.1.4 Positions	26
4.1.5 Templates	27
4.1.6 Main onboarding page	27
4.1.7 New employee functionality overview.....	29
4.2 Web Services	30
5 Development plans	33
5.1 SMTP server and emailing functionality development	33
5.2 Informational security assessments and policy creation.....	33
5.3 Overall architecture layer improvements. Disaster recovery plan	33
5.4 Customize ability level improvements	34
5.5 Candidate progress tracking	34
5.6 Improvement of available Web-services	34
6 Summary.....	35
References	36
Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis	39

List of figures

Figure 1. Overall architecture overview	14
Figure 2. Back end security configuration.....	16
Figure 3. Back-end service configuration	19
Figure 4. YAML configuration file	20
Figure 5. Front end to back end proxy configuration	20
Figure 6. Front end configuration file	21
Figure 7. Database diagram	22
Figure 8. Locations menu	24
Figure 9. Teams menu	25
Figure 10. Users menu.....	26
Figure 11. Positions menu	26
Figure 12. Templates menu	27
Figure 13. Template creation page	28
Figure 14. Template assignment.....	29
Figure 15. Dashboard example view	30

1 Introduction

Proper employee socialization has always been one of the most important aspects. This has a direct positive influence on performance, job satisfaction and organizational commitment of company manpower [1].

Nowadays, in time of digitalization and optimization, more and more companies are in search of a good online platform which would provide new employees a good overview about team, company and learning paths needed to be taken in order to provide all necessary information in an automated framework [2].

Cloud solution developed alongside this thesis is to cover basic needs of a company which is aiming to offer a structured and highly customizable onboarding experience to new employees.

1.1 Main task list

- 1) *Research*. Identify problems and define potential solutions.
- 2) *Technological research*. Identify most suitable software and server solutions for the tool.
- 3) Install all necessary components and jobs setup.
- 4) *Development*. Back end and front end development.
- 5) *Analysis*. Verification of achieved goals.

1.2 Problem statement and objectives

A negative onboarding experience results in new hires being two times more likely to look for other opportunities. Moreover, 88% of employees think their employer did a poor job with the onboarding process. Simultaneously, new employees who went through a

structured onboarding program were 58% more likely to be with the organization after three years [3], [4].

Based on conducted research it was decided to focus on two most common problems currently available onboarding solutions have:

- 1) **Level of customization options:** Multiple solutions lack functionality what leads to HR personnel being unable to offer employees all required information. HR personnel needs to be able to create a template using multiple widgets offered.
- 2) **Integrate ability:** Most mid-sized or large companies have main employee flow automated. This stands for processed to be triggered via REST API. Required is to get all available templates, assign employee to onboarding template, get list of users and get status of onboarding process.

Main goal is to cover basic aspects of both aforementioned problems what would provide HR and an employee a platform to successfully pass the onboarding period and decrease an average retention rate in pandemic time.

2 Technology overview

This chapter provides an overview of technologies, software solutions and software components used for application implementation. Both front end and back end were written using IntelliJ IDEA Ultimate edition.

2.1 Overall architecture

As depicted by Figure 1, architecture layer of developed application follows basic principles of a client-server concept [5].

Instance is hosted in AWS cloud.

A web server is responsible for page rendering and for whole communication flow with a back-end service.

Back-end service is responsible for user authentication and authorization through Spring Security module. Additionally, back end is in control of the data flow and interactions with a database.

A database is hosted on the same server as other application components. The database is storing core objects data of the whole application.

It was decided to keep all application components on one instance, as the goal is to develop a working application. Separate instances will be considered for commercial move to improve stability and performance of the system.

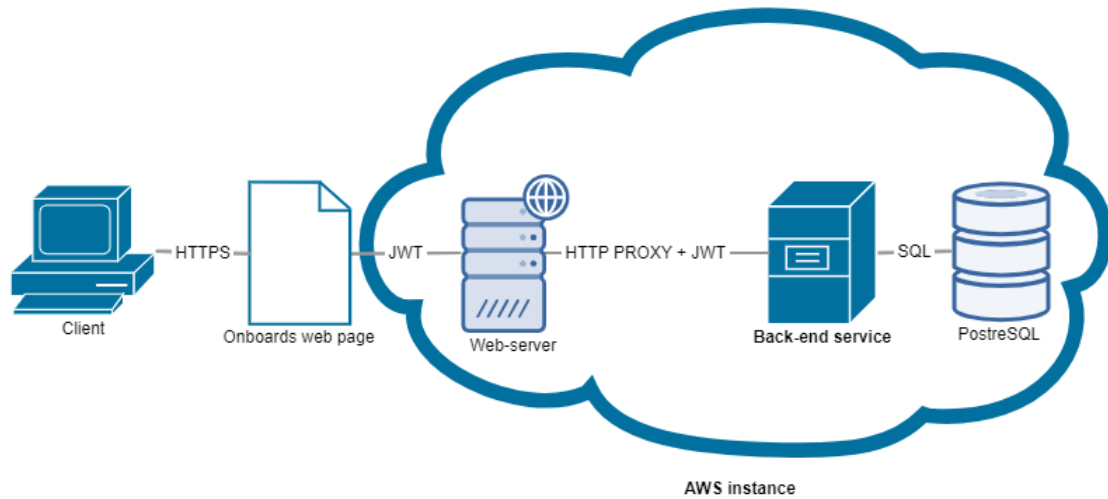


Figure 1. Overall architecture overview

2.2 PostgreSQL

Based on online database ranking and compatibility with Ubuntu AWS instance used for this application the following database management systems were considered for selection: Oracle, MySQL, MongoDB, MariaDB and PostgreSQL [6].

Key components for further selection were: security level, performance, back-up options, back end programming language compatibility, costs, size limitations, efforts needed for configuration and suitability for drafted database model. Based on aforementioned criteria, Oracle and MongoDB were considered as less suitable due to size limitations and huge costs for Oracle and MongoDB not being initially designed for relational data storage. Additional comparison was done and no preferable options were identified. Therefore PostgreSQL was selected because of having biggest amount of experience with this database management system [7], [8], [9], [10].

2.3 Server instance

Amazon Web EC2 web service was selected to be used as a server. More and more organizations move to cloud for multiple reasons including IT operations costs decrease, higher level of security and data protection together with easier customization options through inbuilt tools. As also having no experience with Azure services, AWS was selected [11].

2.4 Back-end technologies

Back end is written using programming language Java with use of Spring Boot framework. Locally project is built using Gradle automation tool.

Built application follows Model-View-Controller software design principles [12].

Below is provided an overview of used technologies.

2.4.1 Lombok

Library is used to reduce boilerplate code. Generates code through annotations processing [13].

2.4.2 JSON Web Token

Standard used to pass identity of authorized users to the server. Implemented alongside Spring Security core configuration. Duration and secret key are defined in *application.yaml* file of the project. Inbuilt classes are used for requests filtering and token generation. Front end is configured to store token of user's session and pass in further request [14].

2.4.3 Spring Security

Open source Java-based framework used to configure authentication, authorization of the application and protect from unauthorized, fraudulent use. In context of developed application works together with *JWT* technology. Main class of the framework defines core security principles used in this application [15].

```

@Override
protected void configure(HttpSecurity http) throws Exception {
    http.csrf().disable()
        .headers().httpStrictTransportSecurity().disable()
        .and()
        .sessionManagement().sessionCreationPolicy(STATELESS)
        .and()
        .exceptionHandling()
        .authenticationEntryPoint(restAuthenticationEntryPoint)
        .and()
        .authorizeRequests()
        .antMatchers("/users/register").permitAll()
        .antMatchers("/users/login").permitAll()
        .anyRequest().authenticated()
        .and()
        .addFilterBefore(jwtRequestFilter,
            UsernamePasswordAuthenticationFilter.class)
        .logout().logoutUrl("/logout");
}

```

Figure 2. Back end security configuration

Is additionally used to define permissions for different roles on a CRUD basis.

2.4.4 Spring Boot

An open source Java-based framework which was selected to simplify the whole configuration and development process through annotations processing [16].

2.4.5 Hibernate and H2 in-memory database

Database tools used to execute configuration locally and verify stability of the data flow and content.

2.5 Front-end technologies

Front end logic is written using Angular. React was selected initially, but due to lack of time and programming experience a switch to Angular was made. A developed application project is enhanced with multiple dependencies, which are used to improve functionality offered to an end-user.

2.5.1 Google maps widget

Google maps widget is connected through an API key and imported dependency. This will be used to visually represent working location to an employee also allowing to check

working place through a Street View functionality. Coordinates will be saved by HR personnel during template creation [17].

2.5.2 Quill text editor widget

Quill *WYSIWYG* text editor is added through a dependency. Main purpose of the widget is to allow HR personnel to create a rich text content which would be displayed to an employee in onboarding form. Quill was selected as is considered to be the best text editor for Angular by multiple rankings [18].

2.5.3 Angular material

Library containing Material Design components for Angular. Used to simplify UI components creation [19].

2.5.4 HTTP interceptor

Inbuilt interceptor used to communicate with a back end. In current project is used to request and response interception, error handling and to request typed response objects [20].

2.5.5 Node Package Manager

NPM was installed locally to fetch dependencies into the project. Is not used on the server side.

2.5.6 Node.js

Node.js is a back-end JavaScript runtime environment which is used to execute JavaScript code locally [21].

3 Server configuration

In the scope of this chapter, an overview is given about installed software on the server side. In addition, this chapter covers configurations done together with an overview of deployment procedures.

3.1 AWS instance creation

New EC2 AWS instance created through a self-management tool with the following parameters: [22]

- 1) Machine image – Free tier Ubuntu Server version 18.04.
- 2) Instance type - t3.micro (free tier eligible).
- 3) Storage – 25GB.
- 4) Security group – Accept all traffic with source value set to 0.0.0.0/0, ::/0
- 5) Public IPv4 address - 13.51.146.191

3.2 JDK and JRE installation

In order to fetch and execute back-end code on the server JDK and JRE, both version 11, were installed on the server.

3.3 Virtual memory

Swap is a space on a disk that is used when the amount of physical RAM memory is full.

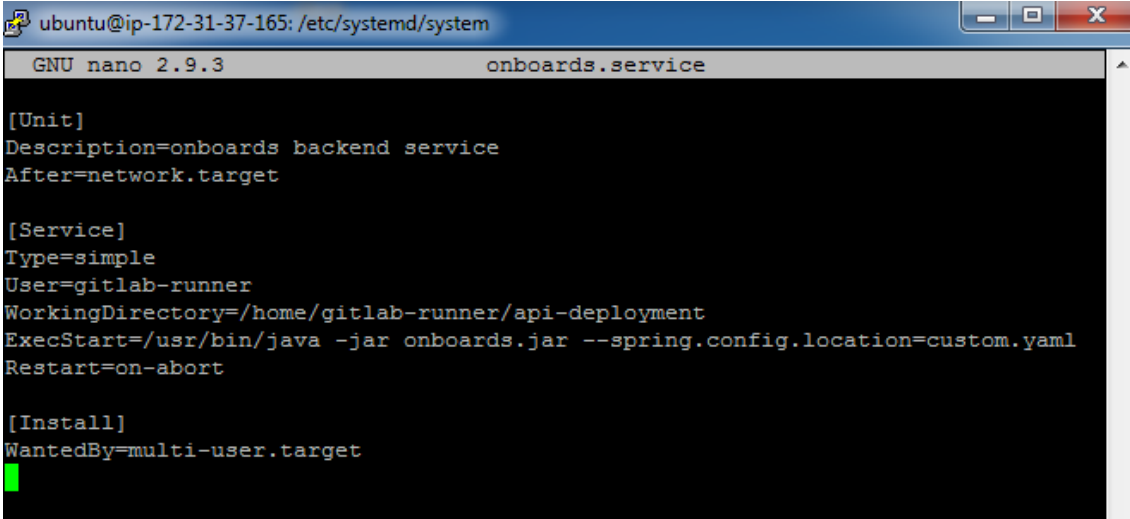
Additional virtual memory was added to improve the performance. This was acknowledged via the creation of a swap file and allocating additional 3 gigabytes to it [23].

3.4 Define back end as a service

In order for the back end to be running on a server under pre-defined custom parameters, a new service was created.

3.4.1 Service configuration

Service is located under `/etc/systemd/system` path and is called `onboards.service`. Configuration of the service is demonstrated below where `onboards.jar` is java archive contains buildable version of back end and which is being executed under custom configuration properties described in `custom.yaml` file [24].



```
ubuntu@ip-172-31-37-165: /etc/systemd/system
GNU nano 2.9.3 onboards.service

[Unit]
Description=onboards backend service
After=network.target

[Service]
Type=simple
User=gitlab-runner
WorkingDirectory=/home/gitlab-runner/api-deployment
ExecStart=/usr/bin/java -jar onboards.jar --spring.config.location=custom.yaml
Restart=on-abort

[Install]
WantedBy=multi-user.target
```

Figure 3. Back-end service configuration

3.4.2 YAML file properties

YAML is a Data-serialization language commonly used for configuration files. In scope of project, a java archive execution property file defines: [24]

- 1) Back end port and context-path.
- 2) Spring Jackson configuration for null values.
- 3) Database connection and configuration properties.
- 4) JWT credentials.
- 5) Path where logs are to be stored.

```
gitlab-runner@ip-172-31-37-165: ~/api-deployment
GNU nano 2.9.3
server:
  port: 8040
  servlet:
    context-path: /api

spring:
  jackson:
    default-property-inclusion: non_null
  jpa:
    database: postgresql
    database-platform: org.hibernate.dialect.PostgreSQLDialect
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        temp:
          use_jdbc_metadata_defaults: false
  datasource:
    driver-class-name: org.postgresql.Driver
    url: jdbc:postgresql://localhost:5432/onboards
    username:
    password:

app:
  jwt:
    secret:
    durationMin: 120

logging:
  path: ./logs
```

Figure 4. YAML configuration file

3.5 NGINX

NGINX was installed to serve the purpose of a web-server. The decision was made based on performance in static context processing.

Created a new configuration under *sites-available* with to *sites-enabled*.

In order to setup proxy to back end configuration *sites-available*, newly created configuration was updated with the following endpoint information:

```
location /api/ {
    proxy_pass    http://localhost:8040;
}
```

Figure 5. Front end to back end proxy configuration

Defined root parameter to front-end deploy folder in order for front end code to become available. Configuration is attached below:

```

root /var/www/front-deployment;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name onboards.tech www.onboards.tech;

location / {
    index index.html index.htm;
    if (!-e $request_filename){
        rewrite ^(.*)$ /index.html break;
    }
}

location /api/ {
    proxy_pass http://localhost:8040;
    # pass PHP scripts to FastCGI server

```

Figure 6. Front end configuration file

3.6 Node.js and Yarn

Node.js and was installed to execute Angular code. Yarn was installed to manage dependencies.

3.7 DNS configuration

A new domain name was purchased and registered at one of the providers to avoid using IP as a hostname. A new hosted zone was created in Amazon Route 53 Hosted zone menu. In order for traffic to be forwarded to a newly created hosted zone *nameservers* domain configuration was adjusted accordingly [25].

3.8 CERTBOT

Certbot was installed for nginx to support HTTPS traffic. A new SSL certificate was generated for previously created domains [26].

3.9 PSQL installation and model

PostgreSQL was installed on Ubuntu with new Onboards database created [27].

The database relational diagram is as follows:

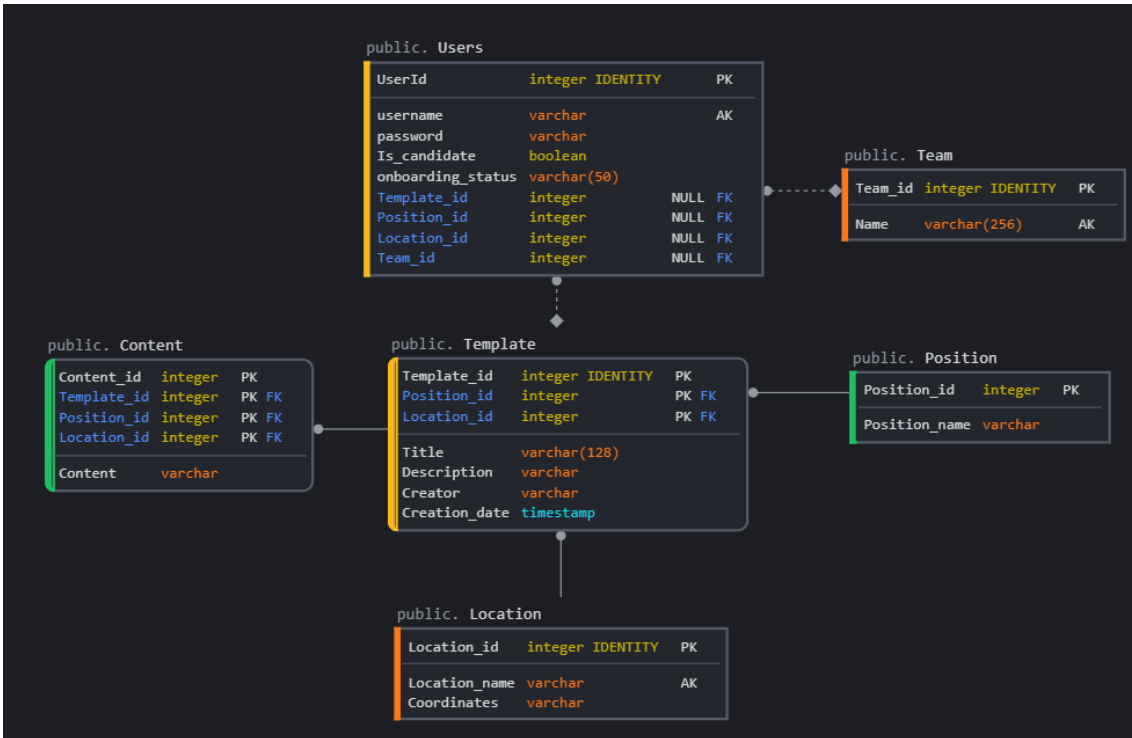


Figure 7. Database diagram

Two core tables are Template and Users. There are no standalone tables in the schema and all the tables are linked with the use of foreign keys. Content of the template is saved in Content, Position and Location tables. Location table contains coordinates used to render Angular Google Maps. Content table content HTML code as a String rendered from Quill text editor. User can have only one template and belong to one team.

3.10 Code build and deployment

In order to make code commits directly to a virtual machine, two gitlab-runners from gitlab.cs.ttu.ee were registered: one for back-end project and the other for front-end. This allows code to be built and pushed directly from used IDE to the server. Executed jobs are configured in `.gitlab-ci.yml` files [24].

The build step defines build parameters, and deploy is responsible code deployment and folder preparations. Same applies to front-end jobs.

4 User interface and REST API

This chapter contains results of development and is to demonstrate what functionality is available corresponding to the problem statement.

4.1 User Interface

Below is a description of all functionality available for application users based on permissions given.

4.1.1 Locations

Locations are used in the system to define future employee working address. Locations' data is stored in the database and is retrieved via Controller on a page initialization.

Locations are represented in two ways:

- 1) Manually entered location address.
- 2) Coordinates defined via Angular Google Maps marker.

Locations can be created and deleted. Locations administrative menu is accessible over `https://onboards.tech/locations` path. Figure 8 demonstrates locations' menu example.

Locations

Create new Location

Location name

Please be precise 0 / 52





	Title	
1	Tallinn, Ehitajate 5	
2	Tallinn, Lootsa 2B	
3	Tallinn, Viru Keskus	
12	Tallin, Liikuri 8A	

Figure 8. Locations menu

4.1.2 Teams

Teams are used in the templates to offer a candidate an option to see future colleagues. As teams' data is built based on information received over API, it also offers a possibility for Admin and HR to verify data correction.

Teams' data is stored in the database and is retrieved via Controller on a page initialization.

No teams can be created and no users assigned via user interface as this is to happen through a web service. This is designed in such way to ensure data quality and stability within the systems. Teams' administrative menu is accessible over <https://onboards.tech/teams> path. Figure 9 demonstrates teams' menu example.

Teams

	Team naming	Users
1	HRIT team	admin Samantha Doe
2	Development team	N/A

Figure 9. Teams menu

4.1.3 Users

Users menu is used to provide an overview of all users existing in the system with information about every user. Besides username it includes:

- 1) User's role.
- 2) User's onboarding status.
- 3) User's template.
- 4) User's team.

Users' data is stored in the database and is retrieved via back-end controller on a page initialization.

For now, users can only be created via self-registration or an API. There is no functionality within user interface for new users' creation. Users can be deleted but modification is done only via API or DB query. Users' administrative menu is accessible over `https://onboards.tech/users` path. Figure 10 demonstrates users' menu example.

Username	Role	Onboarding Status	Template	Team
admin	ADMIN		N/A	HRIT team
tester	CANDIDATE	STARTED	template	N/A
John Deal	CANDIDATE	UNASSIGNED	N/A	N/A
Samantha Doe	CANDIDATE	STARTED	IT template	HRIT team

Figure 10. Users menu

4.1.4 Positions

Positions are linked with templates. Goal is to demonstrate to what position every template belongs. This should help HR to increase a well-structured template database which would eventually allow to assign the same templates multiple times.

Positions data is stored in the database and is retrieved through back-end controller on a page initialization.

Positions can be created and deleted. Positions' administrative menu is accessible over <https://onboards.tech/positions> path. Figure 11 demonstrates positions' menu example.

Create new Position

Position name

Please be precise 0 / 52

Create

Title




1	Java developer	
4	System analyst	
6	Team lead IT	

Figure 11. Positions menu

4.1.5 Templates

Template is the main object of an application. It contains references to multiple other objects created by the administration team which are built together and displayed to a candidate providing all the relevant information for successful onboarding. Currently, templates refer to the following entities:

- 1) Content.
- 2) Location.
- 3) Position.
- 4) Users.

Templates data is saved in the database and retrieved automatically once a page is rendered.

Templates' administrative menu is accessible over <https://onboards.tech/templates> path. Figure 12 demonstrates templates' menu example

Templates

Title	Description	Creator	Creation Date	Location	Position
test template	test template for column generator	Oleg Barinov	14-11-21	Tallinn, Ehitajate 5	Java developer
template	test template for column generator	Oleg Barinov	14-11-21	Tallinn, Lootsa 2B	Java developer
IT template	This template is for IT people	admin	13-05-2021	Tallin, Liikuri 8A	Team lead IT

Figure 12. Templates menu

4.1.6 Main onboarding page

Main menu for template content creation. With use of this functionality users are able to define the content a newly hired employee would see when accessing the system and also to assign a template to an employee.

Templates are created using inbuilt modules. Currently the following modules are available:

- 1) Quill text editor to define a rich HTML content.
- 2) Angular Google Maps widget with a marker.

Additionally, user defines position, title and a template description. The template creation page is accessible over `https://onboards.tech/start` path.

The template content is saved in the database and is referenced to other tables with use of foreign keys. Data is retrieved through Template controller.

Main content creation widget looks in the following way:

template title template description

Please pick unique naming 0 / 52

Enter Welcoming to a candidate


B I U **Normal**

Type here to define Welcoming part - LIMIT of 500 characters applies!

Select working location

Location

Please enter full location address



Choose employee position

Please choose position...

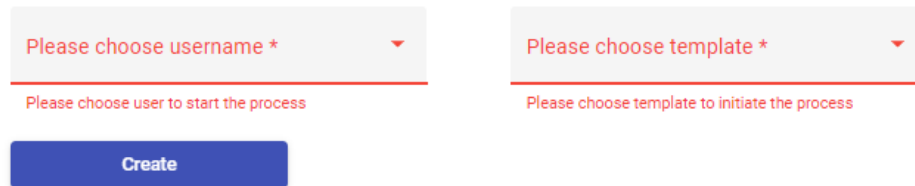
Figure 13. Template creation page

Template is assigned to a user from <https://onboards.tech/start> page.

Template can be only assigned to an employee who does not have the templates assigned.

The assigning page is the following:

Assign existing template to a candidate



Please choose username *
Please choose user to start the process

Please choose template *
Please choose template to initiate the process

Create

Figure 14. Template assignment

4.1.7 New employee functionality overview

Employee onboarding dashboard has all the information previously defined by HR in process of template creation. Dashboard is currently designed to be read-only. Dashboard has the following key elements:

- 1) HTML-based introduction content created by HR in Quill editor.
- 2) Position information.
- 3) Working address with Google Maps widget including location marker. It is used to allow an employee to check working location through Google Streets.
- 4) Team naming and colleagues.

Example dashboard looks in the following way:

Welcome aboard Samantha Doe!

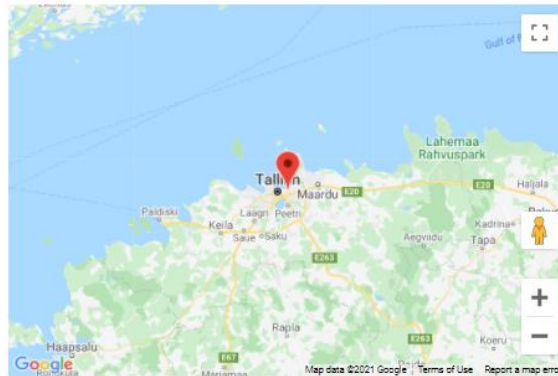
Please see general information below:

Hello and welcome to our IT company!

In order to enjoy your stay here please login to our [WEBSITE](#) and keep on rolling!

This onboarding page is designed for Team lead IT Position

Your working address is Tallin, Liikuri 8A. You can see location on the map below:



Your future team mates:

Your team name - HRIT team

Your colleagues:

admin

Samantha Doe

Figure 15. Dashboard example view

4.2 Web Services

For the purpose of data communication between end user and back end multiple web services were developed. Endpoints can be consumed by other applications in order to create, update, read and delete data. Specific API users will have to be registered with separate token provided. There is no automated solution for access provision therefore it needs to be configured manually.

Below is a list of web services currently available to be consumed:

1) Template:

- a. Get all templates.
- b. Get templates by unique identifier.
- c. Get template by title.
- d. Create new template.
- e. Update existing template.
- f. Delete template.

2) User:

- a. Get all users.
- b. Get user by username.
- c. Get users by team unique identifier.
- d. Register user.
- e. Delete user.
- f. Update existing user.

3) Team:

- a. Get team by unique identifier.
- b. Get all teams.
- c. Delete team.

4) Position:

- a. Get all positions.
- b. Get position by unique identifier.

- c. Delete position.
- d. Create new position.

5) Location:

- a. Get all locations.
- b. Get location by unique identifier.
- c. Delete location.
- d. Create new location.

6) Content:

- a. Get all content.
- b. Get content by unique identifier.
- c. Get content by template unique identifier.
- d. Create new content.

5 Development plans

This chapter covers next application development plan required to step into the market and become more usable and attractive for the customers.

5.1 SMTP server and emailing functionality development

It is important for an application to contain both automated and customizable emails which could be triggered by HR. Automated emails could be:

- 1) Registration confirmation.
- 2) Introduction to onboarding to a candidate once template is assigned.
- 3) Confirmation to HR once template is assigned to a candidate.
- 4) Confirmation to a candidate upon onboarding completion.
- 5) Confirmation to HR upon employee's onboarding completion.

Customizable emails could be used as a functionality for HR to reach out to a candidate with any questions using specific application related tags.

For this purpose SMTP server would need to be configured together with a front-end functionality.

5.2 Informational security assessments and policy creation

For the majority of EU-hosted companies informational security is important therefore multiple assessments would need to be conducted in order to define a level of security and corresponding measures for risks mitigations. Informational Security Policy would need to be created in order to document all security related measures taken to ensure data stability.

5.3 Overall architecture layer improvements. Disaster recovery plan

As of now, database is hosted on the same server with back end and front end what makes application slower and less stable in case of unpredicted downtime. As a first mitigation

step a separate database instance needs to be created. Same applies to disaster recovery plan which needs to be created to document all the steps needed to be taken in case of an unpredicted downtime.

5.4 Customize ability level improvements

Main page of an application contains multiple widgets which make the application more attractive and usable for HR. However, to meet the requirements a constant development and introduction of new modules is in plan. This is to ensure application meets high standards of the customers. One of the main enhancements would be to allow image and other file storage. There are a lot of documents which companies need to have shared with new employees during onboarding therefore this enhancement has critical importance. Additionally, another important enhancement on the road map is to allow HR to add modules based on their need as currently page has all the modules included by default and only once. However, it provides a higher level of customization if it is possible to build a page from scratch and select modules based on the need and requirements.

5.5 Candidate progress tracking

Currently it is not possible to track candidates' progress as page is read-only for them. Ideally, HR would like to track the progress of their employees. This would be possible if a candidate would need to manually confirm passed steps in order to finish the onboarding. This would be simultaneously displayed to HR.

5.6 Improvement of available Web-services

Already developed controllers handle most of the required API requests. However, to provide a completely scalable and robust solution further development needs to be made. This would need to be done together with the integrated partners.

6 Summary

The main purpose of the thesis was to develop an onboarding solution which would cover two biggest problems identified in a market research phase: integrate ability and level of available customizations offered to content creators. Solution developed was intended to be of interest for large and mid-sized companies.

As an outcome of development, a new and fully scalable solution was created with well-known, modern technologies used. This makes solution more flexible from operational perspective.

The main goal of the thesis can be considered as achieved as HR personnel is able to define the content they would like their new employees to see. Moreover, multiple integrations were made possible what should cover main needs of integrating clients.

However, based on the feedback received, level of currently offered customization options is rather small to accommodate all the needs of large companies on the market. Despite having multiple integration endpoints, level of customizations offered remains the main acceptance criteria for most of the companies.

It makes this solution more applicable to small and mid-sized companies, instead. However, as there are concrete development plans defined, it should be significantly easy to improve the tool in order to match expectations of the biggest and most demanding customers on the market.

References

- [1] Wikipedia, "Onboarding," Wikipedia, 24 April 2008. [Online]. Available: <https://en.wikipedia.org/wiki/Onboarding>. [Accessed May 2021].
- [2] H. team, "8 Reasons Onboarding Is Essential," Hireology, 3 December 2020. [Online]. Available: <https://hireology.com/blog/8-reasons-onboarding-is-essential/#:~:text=Through%20onboarding%2C%20an%20organization%20can,actions%20and%20attitudes%20at%20work..> [Accessed May 2021].
- [3] A. S. Hirsch, "Don't Underestimate the Importance of Good Onboarding," 10 August 2017. [Online]. Available: <https://www.shrm.org/resourcesandtools/hr-topics/talent-acquisition/pages/dont-underestimate-the-importance-of-effective-onboarding.aspx#:~:text=New%20employees%20who%20went%20through,percent%20greater%20new%2Dhire%20productivity..> [Accessed May 2021].
- [4] J. Dewar, "10 Employee Onboarding Statistics you Must Know in 2021," Sapling, 4 May 2021. [Online]. Available: <https://www.saplinghr.com/10-employee-onboarding-statistics-you-must-know-in-2021#:~:text=According%20to%20Digitate%2C%20employees%20who,to%20hire%20a%20new%20worker..> [Accessed May 2021].
- [5] Wikipedia, "Client–server model," Wikipedia, 10 September 2013. [Online]. Available: https://en.wikipedia.org/wiki/Client%E2%80%93server_model. [Accessed May 2021].
- [6] M. Hasan, "The 15 Best Database Management Systems for Linux Desktop," 17 April 2021. [Online]. Available: <https://www.ubuntupit.com/best-database-management-systems-for-linux/>. [Accessed May 2021].
- [7] Guru99, "13 BEST Free Database Software (SQL Databases List) in 2021," Guru99, 2021. [Online]. Available: <https://www.guru99.com/free-database-software.html>. [Accessed April 2021].
- [8] DB-engines, "System Properties Comparison MariaDB vs. MySQL vs. PostgreSQL," [Online]. Available: <https://db-engines.com/en/system/MariaDB%3BMySQL%3BPostgreSQL>. [Accessed April 2021].
- [9] K. Gupta, "MARIADB VS. MYSQL VS. POSTGRESQL IN-DEPTH COMPARISON," FreelancingGig, 22 July 2017. [Online]. Available: <https://www.freelancinggig.com/blog/2017/07/22/mariadb-vs-mysql-vs-postgresql-depth-comparison/#:~:text=MariaDB%20used%20ACID%20properties%20for,no%20difference%20between%20the%20platforms.&text=Whereas%20PostgreSQL%20only%20supports%20Master%2D%20Slave%20>. [Accessed April 2021].
- [10] D. Tobin, "Which Modern Database Is Right for Your Use Case?," Xplenty, 10 June 2020. [Online]. Available: <https://www.xplenty.com/blog/which-database/>. [Accessed April 2021].

- [11] D. Linthicum and K. Ramachandran, “Why organizations are moving to the cloud,” Deloitte, 2021. [Online]. Available: <https://www2.deloitte.com/us/en/insights/industry/technology/why-organizations-are-moving-to-the-cloud.html>. [Accessed April 2021].
- [12] “Model–view–controller,” Wikipedia, 1 December 2015. [Online]. Available: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>. [Accessed May 2021].
- [13] baeldung, “Introduction to Project Lombok,” baeldung, 2020. [Online]. Available: <https://www.baeldung.com/intro-to-project-lombok>. [Accessed May 2021].
- [14] I. Gorgadze, “Spring Security with JWT for REST API,” 2020. [Online]. Available: <https://www.toptal.com/spring/spring-security-tutorial>. [Accessed May 2021].
- [15] Spring, “Spring Security,” 2021. [Online]. Available: <https://spring.io/projects/spring-security>. [Accessed May 2021].
- [16] Spring, “Spring Boot,” 2021. [Online]. Available: <https://spring.io/projects/spring-boot>. [Accessed May 2021].
- [17] J. Keung, “Integrating Google Maps API w/ Angular 7+,” 7 February 2019. [Online]. Available: <https://medium.com/@jkeung/integrating-google-maps-api-w-angular-7-e7672396ce2d>. [Accessed May 2021].
- [18] Openbase, “10 Best Angular WYSIWYG Editor Libraries,” Openbase, 2021. [Online]. Available: <https://openbase.com/categories/js/best-angular-wysiwyg-editor-libraries?orderBy=RECOMMENDED&>. [Accessed May 2021].
- [19] Angular, “Angular Material,” Angular, 2020. [Online]. Available: <https://material.angular.io/>. [Accessed May 2021].
- [20] Angular, “Communicating with backend services using HTTP,” Angular, 2020. [Online]. Available: <https://angular.io/guide/http>. [Accessed May 2021].
- [21] Wikipedia, “Node.js,” Wikipedia, 19 October 2019. [Online]. Available: <https://en.wikipedia.org/wiki/Node.js>. [Accessed May 2021].
- [22] Amazon Web Services, Inc, “Step 2: Create your EC2 resources and launch your EC2 instance,” Amazon Web Services, Inc, 2021. [Online]. Available: <https://docs.aws.amazon.com/efs/latest/ug/gs-step-one-create-ec2-resources.html>. [Accessed April 2021].
- [23] A. Prakash, “How to Create and Use Swap File on Linux,” It's FOSS, 29 August 2019. [Online]. Available: <https://itsfoss.com/create-swap-file-linux/>. [Accessed April 2021].
- [24] O. Barinov, “Setup Flow,” 2019. [Online]. Available: <https://gitlab.cs.ttu.ee/Oleg.Barinov/iti0203-2019-back/-/blob/master/Phase-II%20setup%20guide/Setup%20Flow>. [Accessed April 2021].
- [25] Amazon Web Services, Inc, “Making Route 53 the DNS service for a domain that's in use,” 2021 Amazon Web Services, Inc. [Online]. Available: <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/migrate-dns-domain-in-use.html>. [Accessed May 2021].
- [26] H. Virdó, “How To Set Up Let's Encrypt with Nginx Server Blocks on Ubuntu 16.04,” 19 October 2017. [Online]. Available: <https://www.digitalocean.com/community/tutorials/how-to-set-up-let-s-encrypt-with-nginx-server-blocks-on-ubuntu-16-04>. [Accessed May 2021].

- [27] M. Drake and J. Ellingwood, “Установка и использование PostgreSQL на Ubuntu 18.04,” 7 January 2020. [Online]. Available: <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-18-04-ru>. [Accessed April 2021].

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis¹

I Oleg Barinov

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Development of customisable onboarding solution for remote-work environment”, supervised by Sadok Ben Yahia.
 - 1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
 - 1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

18.05.2021

¹ The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.