
**“Kazakh-British Technical University” JSC**
**Faculty of Information Technology**

# REPORT ON INTERNSHIP

**Company:** “Reself”

**Prepared By: Zaitinov Edige**

**Almaty, 2021**

# CONTENT

# Introduction

**Web development** is the work involved in developing a Web site for the Internet. Web development can range from developing a simple single static page of plain text to complex web applications, electronic businesses, and social network services. A more comprehensive list of tasks to which Web development commonly refers, may include Web engineering, Web design, Web content development, client liaison, client-side/server-side scripting, Web server and network security configuration, and e-commerce development. Among Web professionals, "Web development" usually refers to the main non-design aspects of building Web sites: writing markup and coding. Web development may use content management systems (CMS) to make content changes easier and available with basic technical skills.
For larger organizations and businesses, Web development teams can consist of hundreds of people (Web developers) and follow standard methods like Agile methodologies while developing Web sites. Smaller organizations may only require a single permanent or contracting developer, or secondary assignment to related job positions such as a graphic designer or information systems technician. Web development may be a collaborative effort between departments rather than the domain of a designated department. There are three kinds of Web developer specialization: front-end developer, back-end developer, and full-stack developer. Front-end developers are responsible for behavior and visuals that run in the user browser, while back-end developers deal with the servers.

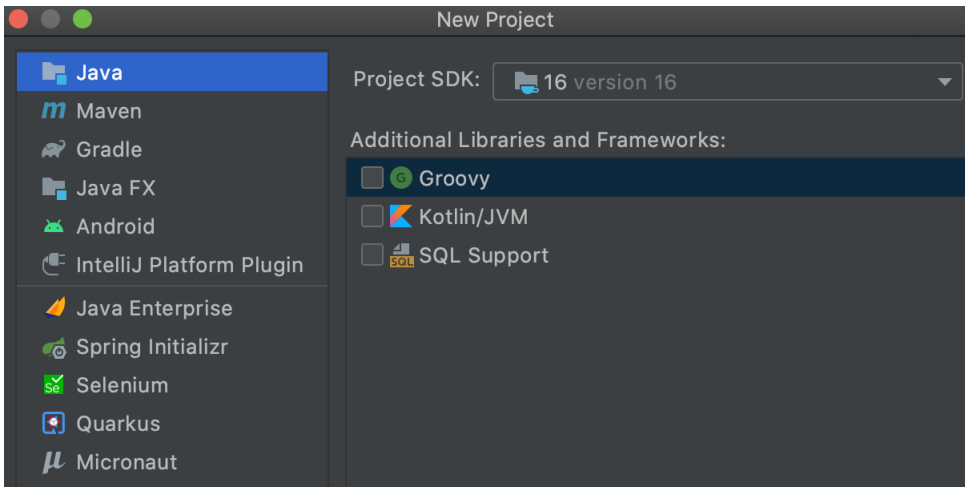
# Intellij Idea

## 1) Introduction to Intellij Idea

IntelliJ IDEA is a smart, context aware IDE. It is designed for developing a variety of applications in Java and other JVM languages such as Kotlin, Scala, and Groovy. In addition, IntelliJ IDEA Ultimate helps you develop web applications with powerful built-in tools, support for JavaScript and related technologies, and advanced support for popular frameworks such as Spring, Spring Boot, Jakarta EE, Micronaut, Quarkus, and Helidon. And free plugins developed by JetBrains allow you to further extend the capabilities of IntelliJ IDEA and use it to work with other programming languages, including Go, Python, SQL, Ruby and PHP.

### Ergonomic environment

IntelliJ IDEA is thought out in every aspect and is ready to use right out of the box. The environment provides quick access to all the features and built-in tools a developer needs, as well as extensive customization options. You can completely customize the environment to suit your workflow: set keyboard shortcuts, install plugins, customize the interface to your liking, and more.




### Deep code analysis

IntelliJ IDEA was created primarily for Java development, but it understands many other programming languages, including Groovy, Kotlin, Scala, JavaScript, TypeScript, and SQL, and offers intelligent assistance in writing code in each of these languages. Initial indexing of the source code allows the IDE to create a virtual map of the project. Using the information from the virtual map, it instantly detects errors, offers context-aware code completion, refactoring, and more.


## Instant navigation and search

IntelliJ IDEA has a wide variety of features to make navigation and searching faster and easier. They help you focus on writing code and work faster.


# Java, Typescrypt

## 1)OOP

Object-oriented programming (OOP) is a programming methodology based on representing a program as a collection of objects, each of which is an instance of a certain class, and the classes form an inheritance hierarchy.

Ideologically, OOP is an approach to programming as to modeling information objects, solving at a new level the main task of structured programming: structuring information from the point of view of controllability, which significantly improves controllability of the modeling process itself, which, in turn, is especially important when implementing large projects.

Manageability for hierarchical systems implies minimizing data redundancy (similar to normalization) and their integrity, so what is created is conveniently manageable - it will be easy to understand. Thus, through the tactical task of controllability, a strategic task is solved - to translate the programmer's understanding of the task into the most convenient form for further use.

The basic principles of structuring in the case of OOP are related to various aspects of the basic understanding of the subject problem, which is required for optimal control of the corresponding model:

abstraction for highlighting in the simulated subject that is important for solving a specific problem in the subject, ultimately - the contextual understanding of the subject, formalized in the form of a class;

encapsulation for fast and safe organization of hierarchical controllability itself: so that a simple "what to do" command is sufficient, without simultaneously specifying how to do it, since this is a different level of control;

inheritance for fast and safe organization of related concepts: so that it would be enough to take into account only changes at each hierarchical step, without duplicating everything else taken into account in the previous steps;

polymorphism to determine the point at which it is better to parallelize a single control, or vice versa - to put it together.

## 2) Spring Boot

Spring Boot is a useful project that aims to make it easier to build Spring-based applications. It allows you to create a web application in the easiest way, requiring developers to have a minimum of effort to configure and write code.

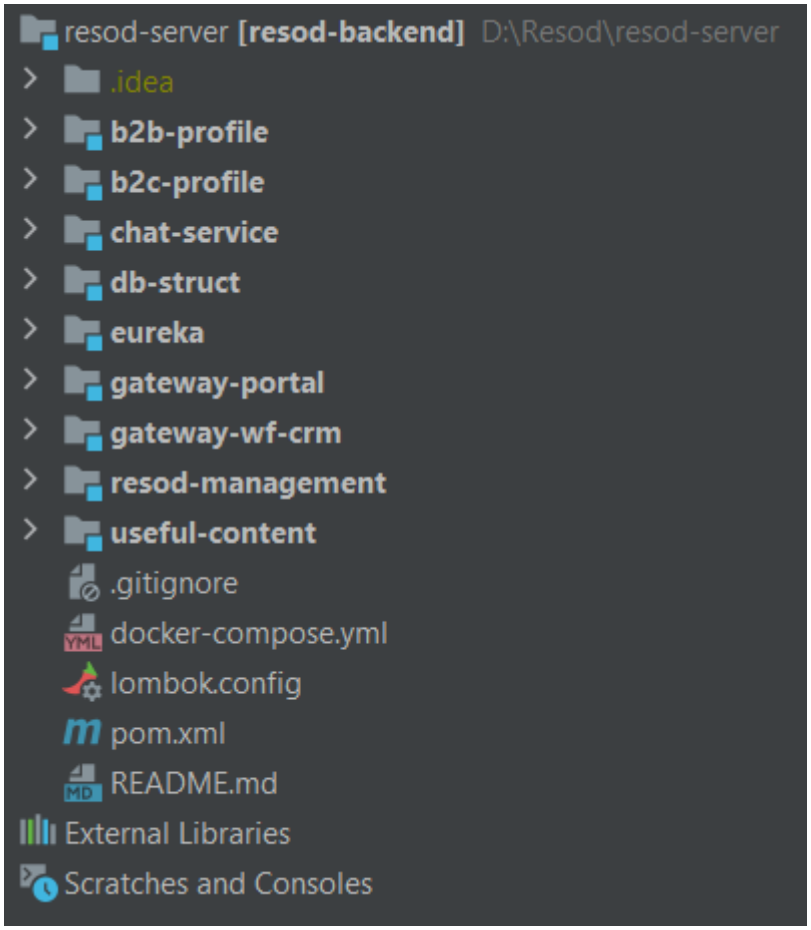
Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

Features

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' dependencies to simplify your build configuration
- Automatically configure Spring and 3rd party libraries whenever possible
- Provide production-ready features such as metrics, health checks, and externalized configuration
- Absolutely no code generation and no requirement for XML configuration


```java
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class DemoApplication {

   public static void main(String[] args) {
      SpringApplication.run(DemoApplication.class, args);
   }

   @GetMapping("/hello")
   public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
      return String.format("Hello %s!", name);
   }
}
```

The backend of our site consisted of several modules. They were responsible for different tasks.




DbStruct was responsible for the structure of the Database.Inside dbStruct module we wrote models  and repositoryes for them. Gateway-wf-crm was responsible for the authorization and authentication in CRM. Resod-management was responsible for functionality. Inside resod-management we wrote controllers and services for them.

## 3) Angular

Angular is an open-source and free web application development platform written in TypeScript, developed by a team at Google, as well as a community of developers from various companies. Angular is a completely rewritten framework from the same team that wrote AngularJS.

## DEVELOP ACROSS ALL PLATFORMS

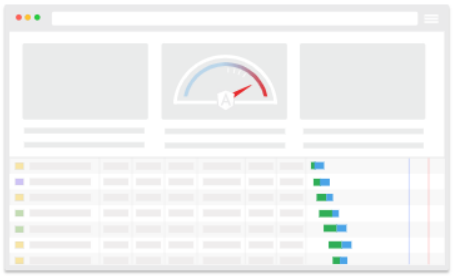
Learn one way to build applications with Angular and reuse your code and abilities to build apps for any deployment target. For web, mobile web, native mobile and native desktop.

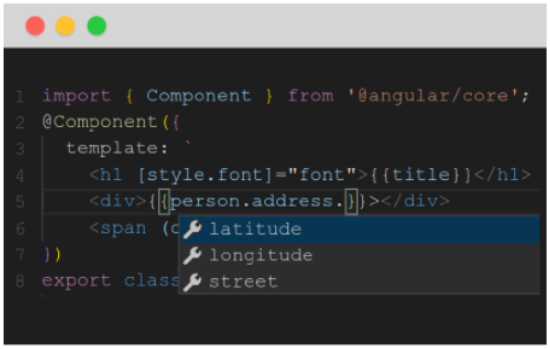
---

## SPEED & PERFORMANCE

Achieve the maximum speed possible on the Web Platform today, and take it further, via Web Workers and server-side rendering.

Angular puts you in control over scalability. Meet huge data requirements by building data models on RxJS, Immutable.js or another push-model.



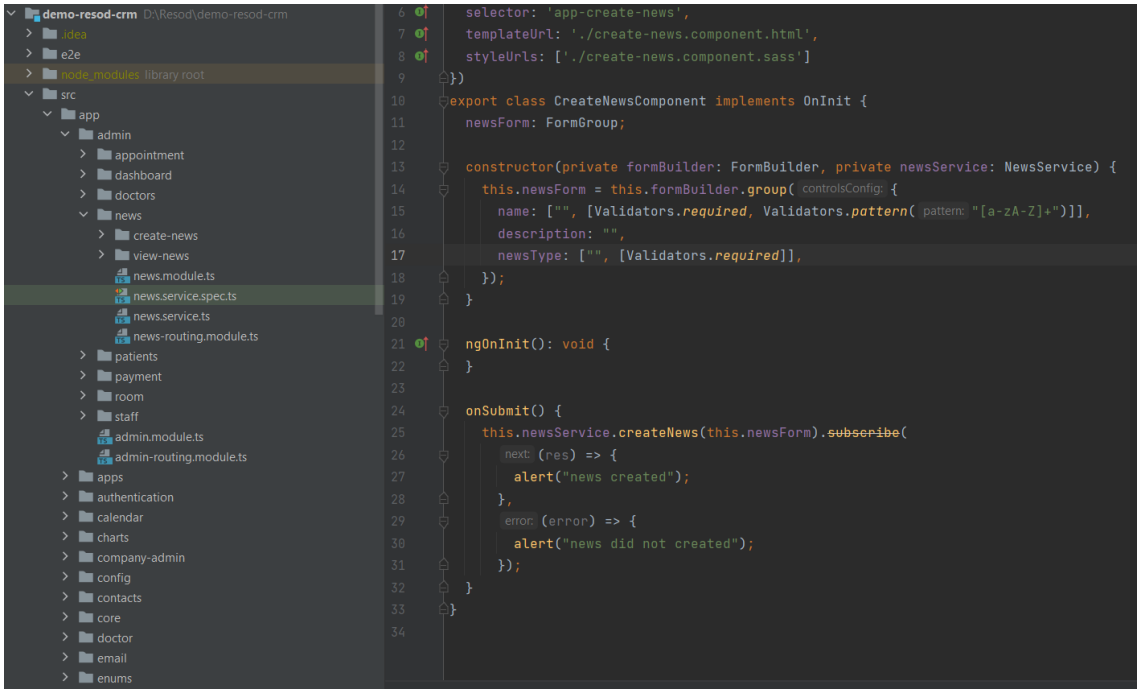



## INCREDIBLE TOOLING

Build features quickly with simple, declarative templates. Extend the template language with your own components and use a wide array of existing components. Get immediate Angular-specific help and feedback with nearly every IDE and editor. All this comes together so you can focus on building amazing apps rather than trying to make the code work.

During development, I was given various tasks from page layout to writing component functionality.

**one of the services of my project**




# Design

## 1) HTML templates

Each Angular template in your application is a section of HTML to include as a part of the page that the browser displays. An Angular HTML template renders a view, or user interface, in the browser, just like regular HTML, but with a lot more functionality.

When you generate an Angular application with the Angular CLI, the `app.component.html` file is the default template containing placeholder HTML.

The template syntax guides show you how to control the UX/UI by coordinating data between the class and the template.

```
            <h4 class="page-title">Book Appointment</h4>
          </li>
          <li class="breadcrumb-item bcrumb-1">
            <a routerLink="/admin/dashboard/main">
              <i class="fas fa-home"></i> Home</a>
          </li>
          <li class="breadcrumb-item bcrumb-2">
            <a href="#" onClick="return false;">Appointment</a>
          </li>
          <li class="breadcrumb-item active">Book</li>
        </ul>
      </div>
    </div>
  </div>
<div class="row clearfix">
  <div class="col-xl-12 col-lg-12 col-md-12 col-sm-12">
    <div class="card">
      <div class="header">
        <h2>Book Appointment</h2>

      </div>
      <div class="body">
        <form class="m-4" [formGroup]="bookingForm" (ngSubmit)="onSubmit()">
          <div class="row">
            <div class="col-xl-6 col-lg-6 col-md-12 col-sm-12 mb-3">
              <mat-form-field class="example-full-width mb-3">
                <mat-label>First name</mat-label>
                <input matInput formControlName="first" required>
                <mat-error *ngIf="bookingForm.get('first').hasError( errorCode: 'required')">
```

Angular helps you get and set DOM (Document Object Model) values dynamically with features such as built-in template functions, variables, event listening, and data binding.

Almost all HTML syntax is valid template syntax. However, because an Angular template is part of an overall webpage, and not the entire page, you don't need to include elements such as `<html>`, `<body>`, or `<base>`, and can focus exclusively on the part of the page you are developing.

## 2) CSS

CSS is used by web page creators to specify colors, fonts, styles, block placement, and other aspects of the presentation of the appearance of these web pages. The main goal of CSS development was to separate the description of the logical structure of a web page (which is produced using HTML or other markup languages) from the description of the appearance of that web page (which is now produced using the formal language CSS). This separation can increase the accessibility of a document, provide more flexibility and control over its presentation, and reduce complexity and repetition in structured content.

Angular applications are styled with standard CSS. That means you can apply everything you know about CSS stylesheets, selectors, rules, and media queries directly to Angular applications.

Additionally, Angular can bundle *component styles* with components, enabling a more modular design than regular stylesheets.

This page describes how to load and apply these component styles.

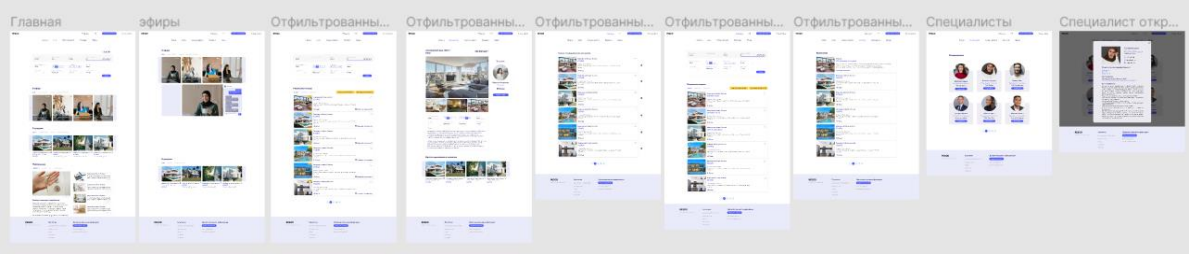
```
.home_container {
  position: relative;
  width: 100%;
  min-height: 100%;
}

h1, h2, h3, h4, h5, h6 {
  color: #231D64;
}

.in_streams_container {
  margin-top: 90px;
}

h6.stream_label {
  margin-left: 40px;
  font-style: normal;
  font-weight: 900;
  font-size: 22px;
  line-height: 30px;
  color: #231D64;
}

.video_wrapper {
  position: absolute;
```

## 3) Figma

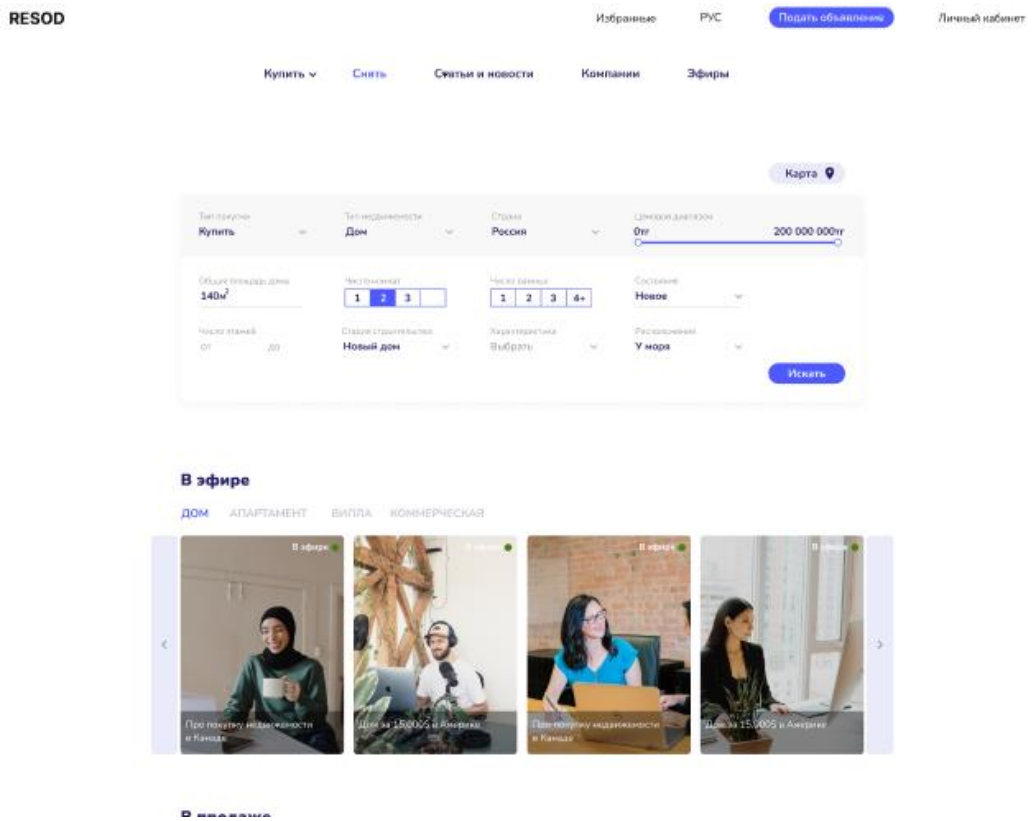
## The entire design of our project was developed in Figma.

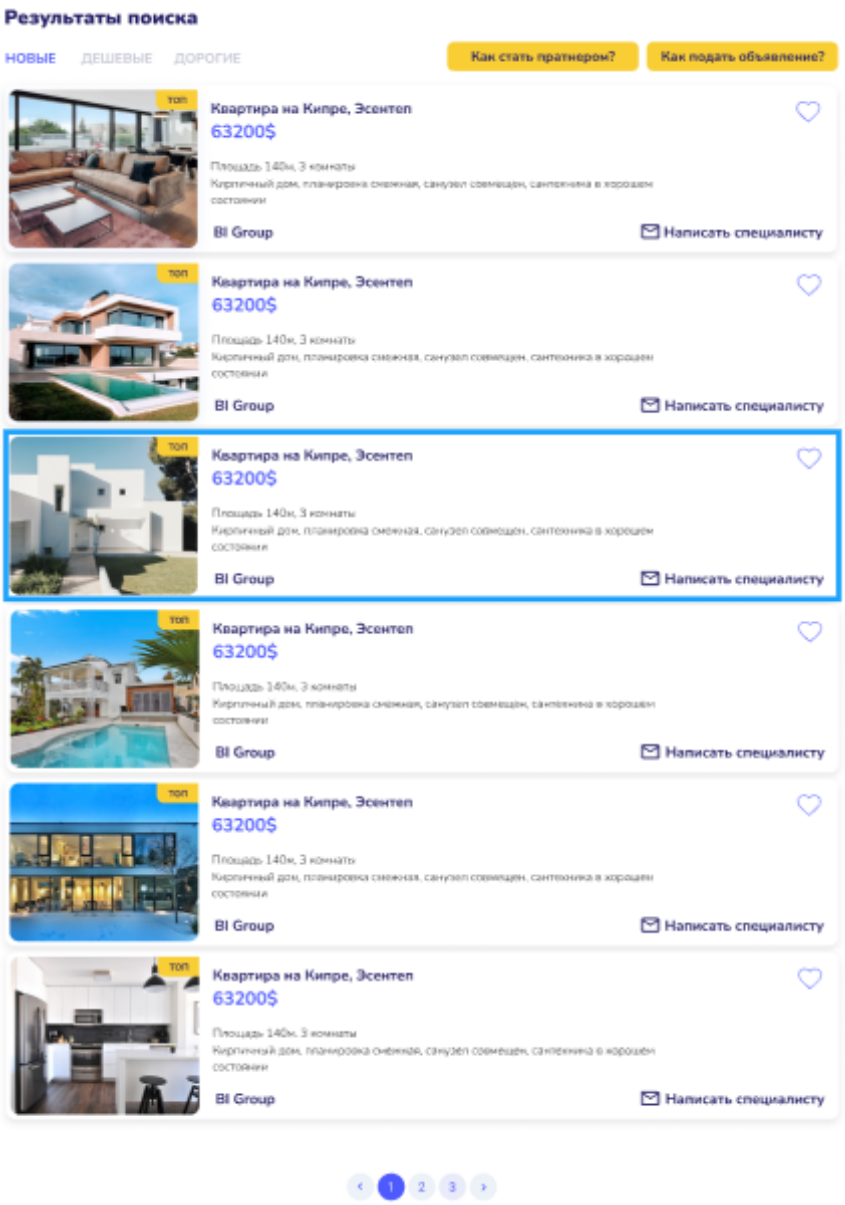
Figma is an online service for interface development and prototyping with the ability to organize real-time collaboration.

The service is available by subscription; there is a free tariff plan for one user. There are offline versions for Windows, macOS. Integration with corporate Slack messenger and Framer prototyping tool has been implemented. It is used both for creating simplified interface prototypes and for detailed design of interfaces for mobile applications, websites, corporate portals.




**I made a large web of these pages, the rest were made by my colleagues**






## 4) Angular material design

This is a library specifically for Angulyar. It helps to facilitate the layout process by using ready-made components. And for adaptability we used Angular flex layout

**High quality**

Internationalized and accessible components for everyone. Well tested to ensure performance and reliability.

Straightforward APIs with consistent cross platform behavior.

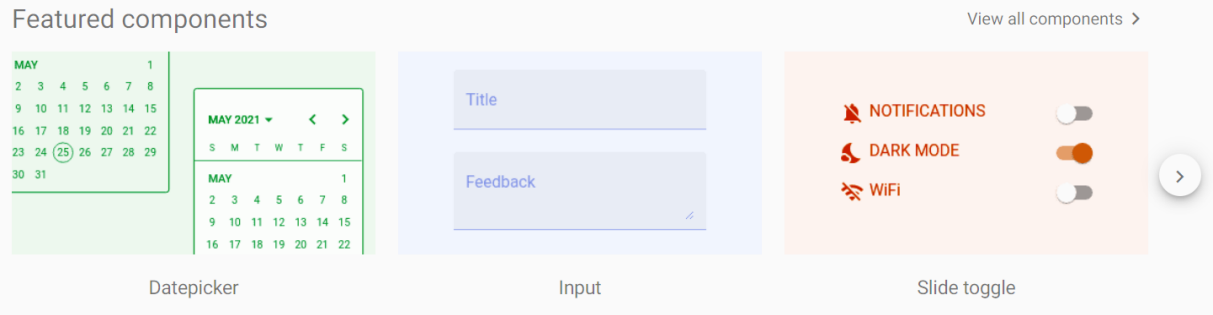
**Versatile**

Provide tools that help developers build their own custom components with common interaction patterns.

Customizable within the bounds of the Material Design specification.

**Frictionless**

Built by the Angular team to integrate seamlessly with Angular.




# Architecture of programming

## 1) Reactive programming Web flux

The WebFlux module appeared in the 5th version of the Spring framework. This micro-framework is an alternative to Spring MVC and represents a reactive approach for writing web services. At the heart of WebFlux is the Project Reactor library, which makes it easy to program non-blocking (asynchronous) streams that handle data input / output.
To develop a reactive service, it is enough to add the spring-boot-starter-webflux module to pom.xml, instead of the usual spring-boot-starter-web in Spring MVC.

Web flux is used in Gateway-wf-crm module.

```
springbootwebflux:
  jjwt:
    password:
      encoder:
```

```
@RequestMapping(value = "/signup", method = RequestMethod.POST)
public Mono<ResponseEntity<?>> createPerson(@RequestBody Empdata user) {
    user.setPassword(passwordEncoder.encode(user.getPassword()));
    return Mono.just(ResponseEntity.ok(userService.addUpdateUser(user)));
}
@PutMapping("/user/{id}")
public Mono<ResponseEntity<?>> updateUser(@PathVariable("id") Long id, @RequestBody Emp
    try {
        if (userService.addUpdateUser(user) != null) {
            return Mono.just(ResponseEntity.ok(userService.addUpdateUser(user)));
        } else {
            return Mono.just(new ResponseEntity<>(HttpStatus.NOT_FOUND));
        }
    } catch (Exception e) {
        System.out.println(e);
        return Mono.just(new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR));
    }
}

@GetMapping("/user/{id}")
public Mono<Empdata> getById(@PathVariable("id") Long id) {
    try {
        return userService.findById(id);
    } catch (Exception e) {
        System.out.println(e);
        return null;
    }
}
```

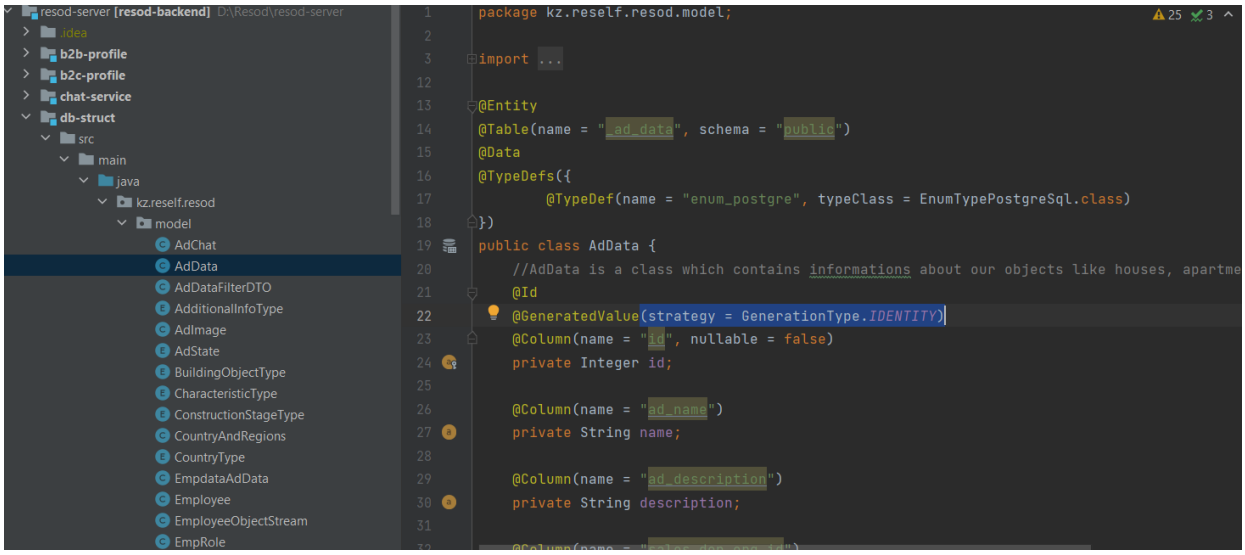
Development using Web Flux is slightly different but gives an advantage in the future

## 2) MVC

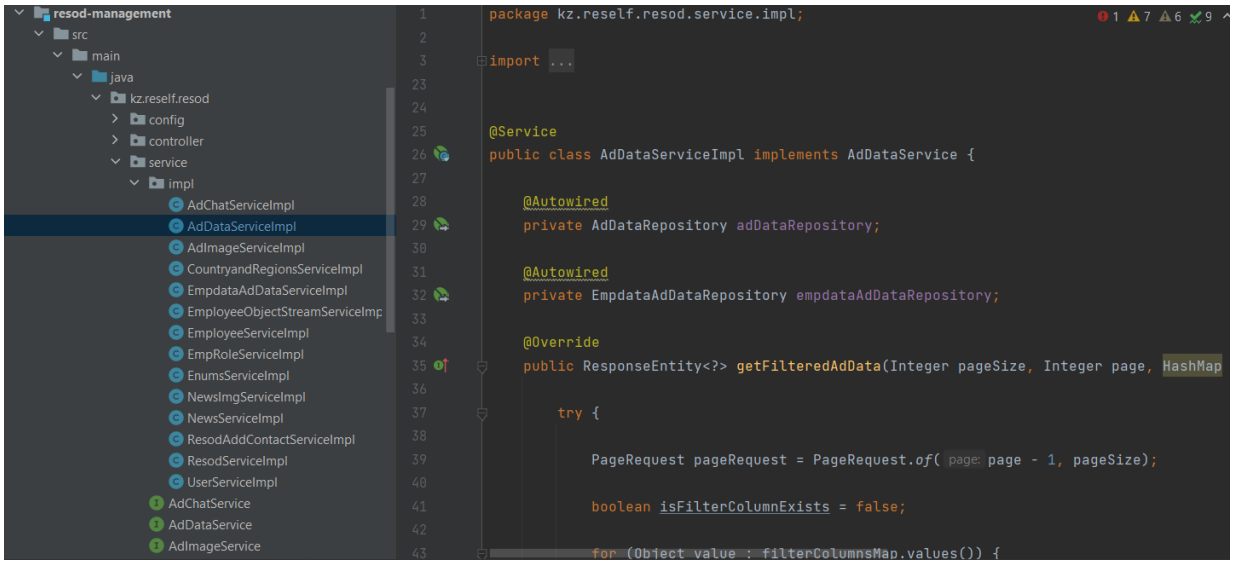
Model-View-Controller (MVC, "Model-View-Controller", "Model-View-Controller") is a scheme for dividing application data and control logic into three separate components: model, view and controller - in such a way that modification of each component can be carried out independently

The Model provides data and reacts to controller commands by changing its state
The View is responsible for displaying model data to the user in response to model changes
The Controller interprets the user's actions, notifying the model about the need for changes.

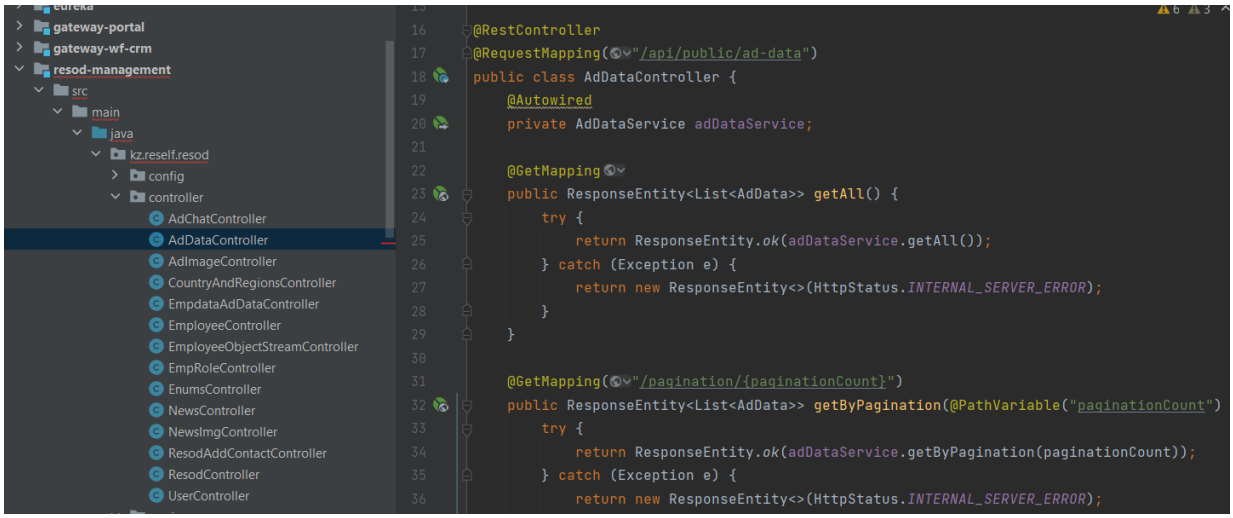
**Model**




**View**



## Controller




# References

1) Intellij idea https://ru.wikipedia.org/wiki/IntelliJ_IDEA

2) Spring boot https://spring.io/projects/spring-boot

3) Angular https://angular.io/

4) Web Flux https://docs.spring.io/spring-framework/docs/current/reference/html/web-reactive.html

5) Angular Material https://material.angular.io/

6) My project is a real project being developed for a company. its essence is in the real estate trade, but with such a feature that it contains video broadcasts. It is still under development.