

Grindin’

Table of Contents

[I. About me 3](#_Toc202301163)

[II. The idea of the project 3](#_Toc202301164)

[III. Stages of work 5](#_Toc202301165)

[Backend Foundations 5](#_Toc202301166)

[Angular Setup & Authentication 5](#_Toc202301167)

[User-Side Functionality 5](#_Toc202301168)

[Admin Features & Preparation 5](#_Toc202301169)

[IV. Used Technologies 6](#_Toc202301170)

[V. Block Scheme 7](#_Toc202301171)

[VI. Entity-Relationship Diagram 8](#_Toc202301172)

[VII. Installation (to run the source code) 9](#_Toc202301173)

[1. Prerequisites 9](#_Toc202301174)

[2. Clone the Repository 9](#_Toc202301175)

[3. Backend Setup (ASP.NET Core) 9](#_Toc202301176)

[4. Frontend Setup (Angular) 10](#_Toc202301177)

[5. Admin Setup 11](#_Toc202301178)

[6. Notes 11](#_Toc202301179)

[VIII. Pictures of UI 12](#_Toc202301180)

[ Login Form 12](#_Toc202301181)

[ Registration Form 12](#_Toc202301182)

[ Home Page 13](#_Toc202301183)

[ Profile Page 13](#_Toc202301184)

[ Job Ads Page 14](#_Toc202301185)

[ Job Applications Page 14](#_Toc202301186)

## About me

|  |  |  |
| --- | --- | --- |
| Name | Role | Grade / Degree |
| Bozhidar A. Dimov | Project Creator and Full-Stack Developer | 11 V / Applied programming |

## The idea of the project

**Grindin'** is a modern and responsive web application designed to streamline the process of tracking, managing, and reviewing job applications. Developed using Angular and TypeScript on the frontend and powered by .NET on the backend, the platform provides a user-friendly interface for job seekers and administrators alike. Whether you're applying to multiple companies or managing submitted applications, **Grindin'** simplifies the job search experience and helps users stay organized and efficient.

The application allows users to:

* Register and manage personal profiles
* Browse and view detailed job advertisements
* Submit applications directly through the platform
* Track application statuses
* Access personalized dashboards
* Create and manage job postings (admin functionality)
* Review and evaluate submitted applications (admin)

With role-based access control, robust form validation, and modern UI components, **Grindin'** offers a secure, accessible, and intuitive experience for both regular users and administrators. The clean separation of modules and the use of Angular best practices ensure maintainability and scalability for future development.

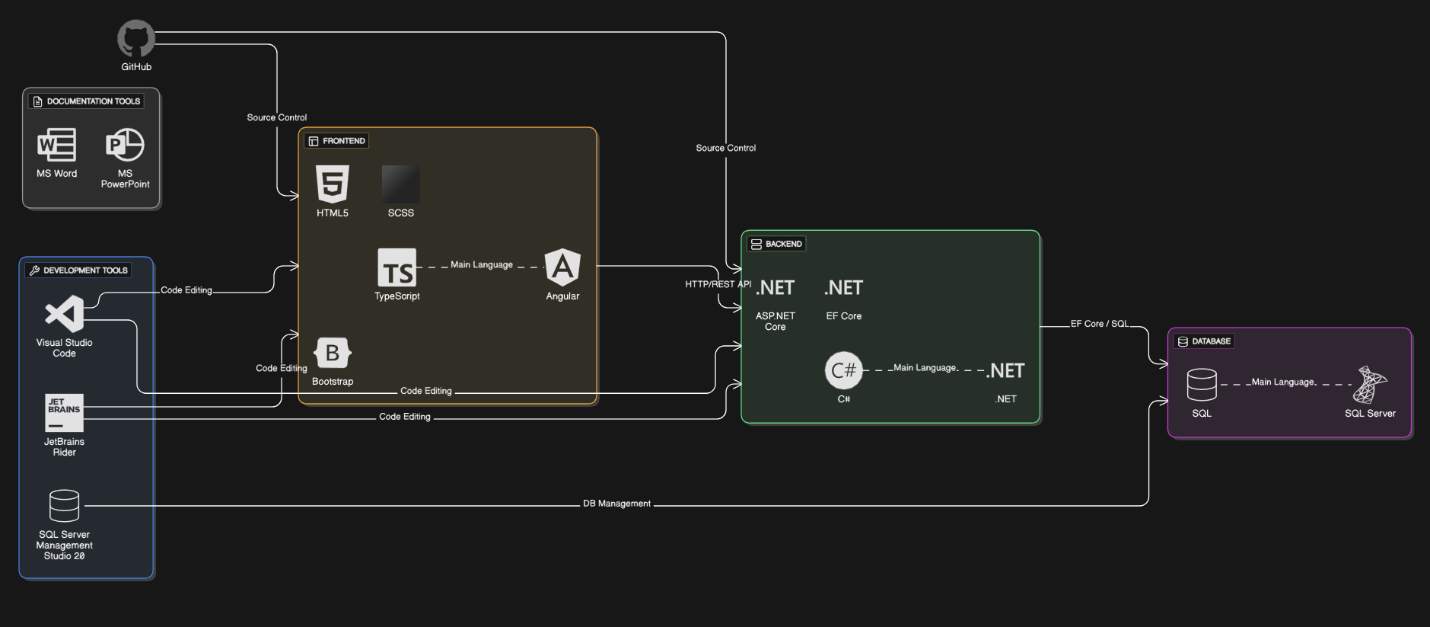
This project was developed as part of the Production Practice curriculum (2024–2025) in class 11 V at the Vocational School of Computer Programming and Innovation (VSCPI) in Burgas, Bulgaria. It serves as both a practical demonstration of software engineering skills and a full-stack web development project, showcasing technologies such as Angular, ASP.NET, Bootstrap, and RESTful API integration.

**Grindin'** reflects the application of theoretical knowledge into a real-world scenario, emphasizing good coding standards, user-centric design, and maintainable architecture. It was completed under the guidance of academic mentors as a milestone in the professional development of the students involved.

## Stages of work

|  |  |
| --- | --- |
| № | Stages |
| **1st Sprint** | Backend Foundations Designed the database models, implemented 3-layer architecture, and built a fully functional API with filtering, DTOs, and services for users, job ads, and applications. |
| **2nd Sprint** | Angular Setup & Authentication Set up the Angular project, built the component structure, designed login and registration forms, and implemented authentication with role-based route guards. |
| **3rd Sprint** | User-Side Functionality Developed user features for browsing and filtering job ads, submitting applications, viewing personal data, and ensured proper frontend validation and navigation. |
| **4th Sprint** | Admin Features & Preparation Implemented admin tools for managing job ads and reviewing applications, added GUI enhancements, and ensured full role-based access control throughout the app and that it functions properly. Enhanced the repository and created the presentation. |

## Used Technologies

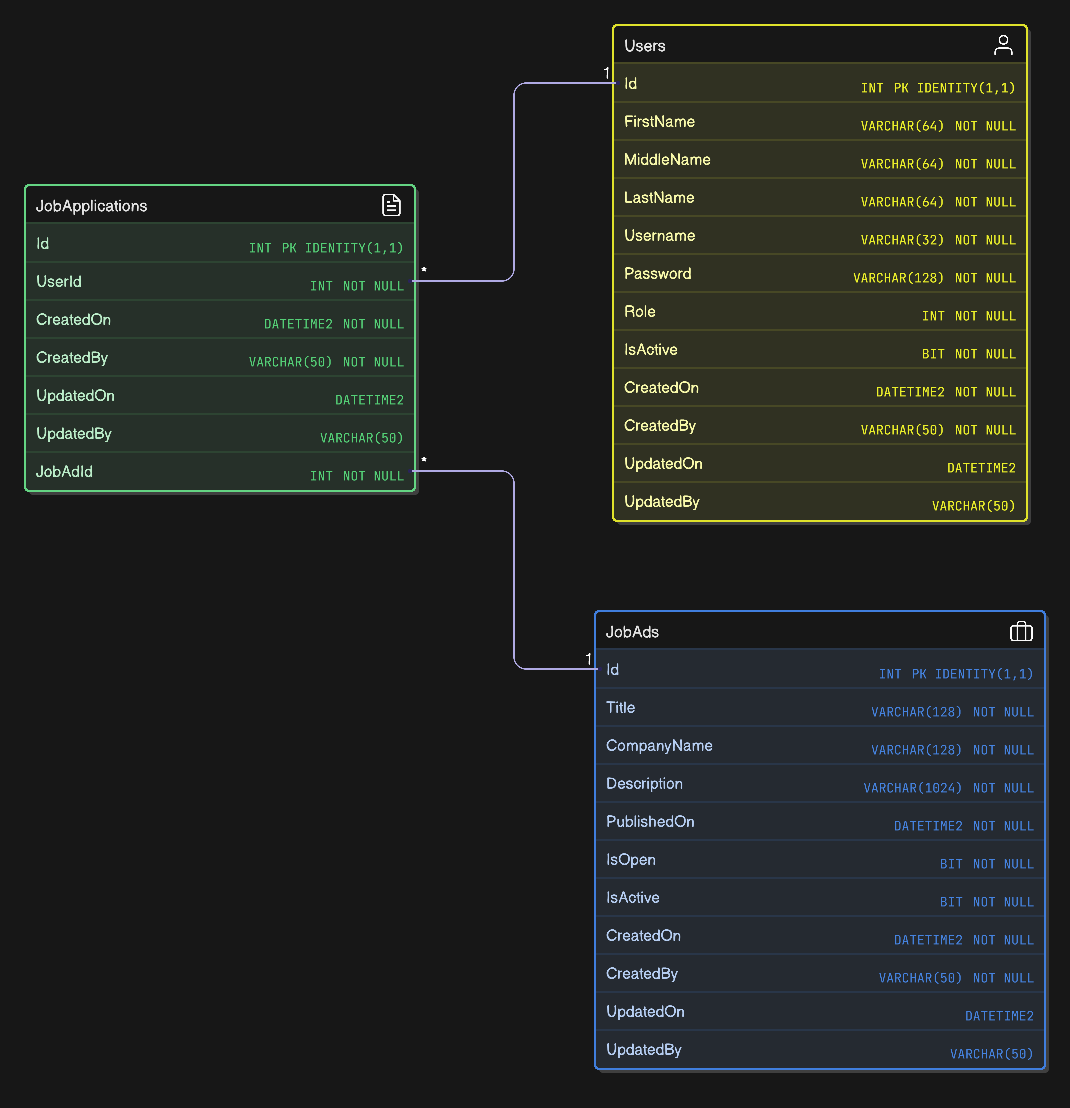


## Block Scheme

A computer screen shot of a diagram

AI-generated content may be incorrect.

## Entity-Relationship Diagram



## Installation (to run the source code)

### Prerequisites

Make sure the following software is installed on your system:

System Tools: Git, SQL Server (Express or Developer), SQL Server Management Studio (SSMS)

Backend: .NET 8 SDK

Entity Framework Core tools:

Run in terminal:

**dotnet tool install --global dotnet-ef**

Required EF Core NuGet packages (already included in the project, but install if needed):

**dotnet add package Microsoft.EntityFrameworkCore.SqlServer**

**dotnet add package Microsoft.EntityFrameworkCore.Design**

Frontend: Node.js (LTS version), Angular CLI (install globally): **npm install -g @angular/cli**

### Clone the Repository

**git clone "https://github.com/codingburgas/job-tracking-BADimov21.git"**

**cd job-tracking-BADimov21**

### Backend Setup (ASP.NET Core)

Navigate to the backend project folder:

**cd JobTracking.API**

After you install MSSQL Server, connect to your local instance using Windows authentication:

**(localdb)\MSSQLLocalDb**

Create the database manually in SSMS or by running this SQL command:

**CREATE DATABASE JobTracking;**

Apply migrations and generate database tables:

**dotnet ef database update**

Run the backend server:

**dotnet run**

The API should now be running at: **https://localhost:5230**

### Frontend Setup (Angular)

Open a new terminal and navigate to the Angular project folder:

**cd ../JobTracking.WEB**

Install dependencies:

**npm install**

Run the frontend server:

**ng serve**

The app will now be accessible at: **http://localhost:4200**

### Admin Setup

Admin users must be manually added to the database.

Set their role to **1** in the *Users* table.

Only admin users can: Create/Edit/Delete job ads; review and manage job applications.

### Notes

Role-based authentication and route protection are implemented on both frontend and backend.

Error handling for bad requests and server issues is implemented using alert messages and user feedback.

## Pictures of UI

### Login Form



### Registration Form

A screen shot of a login form

AI-generated content may be incorrect.

### Home Page

A person standing in front of a blue and white screen

AI-generated content may be incorrect.

### Profile Page

A screenshot of a computer

AI-generated content may be incorrect.

### Job Ads Page

A screenshot of a computer

AI-generated content may be incorrect.

### Job Applications Page

A screenshot of a computer

AI-generated content may be incorrect.

**Version History**

| **Version** |  |  | **Author** | **Date** | **Description** |
| --- | --- | --- | --- | --- | --- |
| 1.0 |  |  | Bozhidar Dimov | 02.07.2025 | Initial documentation |

*End of documentation.*