

TALLINNA TEHNIKAÜLIKOOL

Tallinn University of Technology

Andrei Pestov IADB222764

Smartpost Itella Web App for courier

Web Apps with

C# project

Supervisor: Andres Käver

Tallinn 2024

1 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: Andrei Pestov [25.02.2024]

Table of contents

1	Author's declaration of originality	2
2	Introduction	4
3	Overview	5
3.1	Web interface	5
4	Analysis	6
4.1	ERD model	6
4.2	Database analysis	7
4.3	Messaging and chatting	7
4.4	Courier workflow	7
4.5	User Interface	8
4.6	Project implementation.....	9
5	Summary	10
6	Sources used.....	11
7	Extras.....	12

2 Introduction

In the fast-paced world of parcel delivery, efficiency and accuracy are paramount. Traditionally, parcel couriers in Itella have navigated their routes and managed deliveries using manual processes, relying on Excel tables with all necessary information in them. Couriers use only static Excel tables to look at the locker's location, rounds they need to drive and write manually all the info of their workday including car numbers, comments, and work hours. Although the courier's work is mostly independent, some couriers need to stay in touch during the workday to optimize workflow. Recognizing the need for a more streamlined, modern, and sophisticated solution, the author considers the possibility of improving the system by creating SmartPost Itella Web App for Couriers.

This documentation serves as a comprehensive guide to empower couriers and logistics professionals to adopt a modern and efficient approach to their daily operations.

The purpose of this work is to create a web application aimed to simplify courier's workflow by eliminating the limitations and challenges posed by conventional spreadsheet-based systems. To fulfill this work, the author will create an ERD schema and a database according to courier's needs.

3 Overview

The web app is aimed to optimize the courier's workday, so for that, the author needs to create a fast, responsive web application with a possibility of a chatting, making changes to the courier's workday from his logic operator, reviewing the round/trip/car info. The user will be able to watch and/or manage the trip information according to the events happening during the day.

In common, the app includes a login system, chat between users and a possibility of manipulating user's statistics. The user consists of two main roles: superior and courier. Superiors are allowed to manage manually the lockers and create rounds for couriers, see work hours for each courier, add new employees and manage their roles accordingly. Courier role consists of a shortened version of superior role, which restricts, for example, to change the round, allowing them only to watch the content of the ring and write down comments about the car they drive during the day. Chat possibility is equal to both roles. Each user can write to any other one.

3.1 Web interface

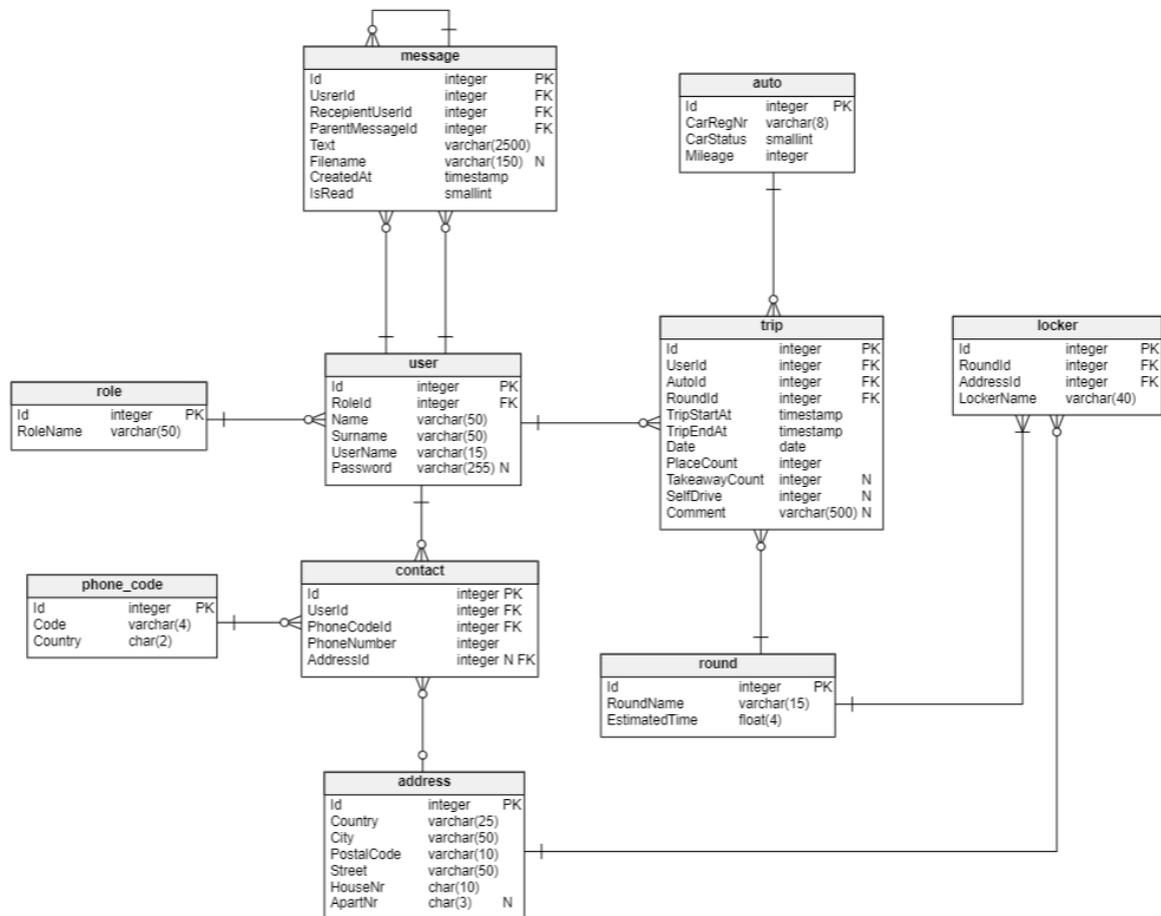
As soon as it comes to web, its interface needs to be minimalistic, intuitive and simple from the user perspective to:

1. Not wasting time.
2. Manage this time effectively.
3. Get the necessary information shortly.

Since all the couriers have only work phones with them, but the superiors work using PC's, the app should also be responsive to the screens of a big and small size.

4 Analysis

4.1 ERD model



Pic. 1: ERD schema

4.2 Database analysis

Since author uses C# MVC controllers, identity (user, role) entities were made just for ERD model clear view and to point out main attributes and relationships needed for the database to work correctly.

4.3 Messaging and chatting

The main point of chatting is to create a simple communication tool between couriers. In the app scope, the user can send another user a message, which has its own text, can be marked as read/unread and the text can also be linked with some files, which user chooses by himself. The main problem here is storing the external files coming from the user. In the app controller author could manage the coming file format by restricting some formats to be uploaded. So, the database should save only the names of the files, but not the files themselves. These files as a best practice can be stored on an external server or (which author prefers more) cloud drive.

4.4 Courier workflow

The courier workflow is concentrated in the auto, trip, round, and locker entities. All the locker's locations are predefined already and available on the Itella website as an open Web API, that's why author had to decide whether to add "locker" entity to database or not. The problem is that API data is changing over time, and it could be a problem with updating it in case API attributes become different. From this perspective the author decided to add the entity to the database, just to make some backup for the API data. In case parsing, deserialization etc. fail app will always have a backup data in database just to not stop the entire courier's teamwork.

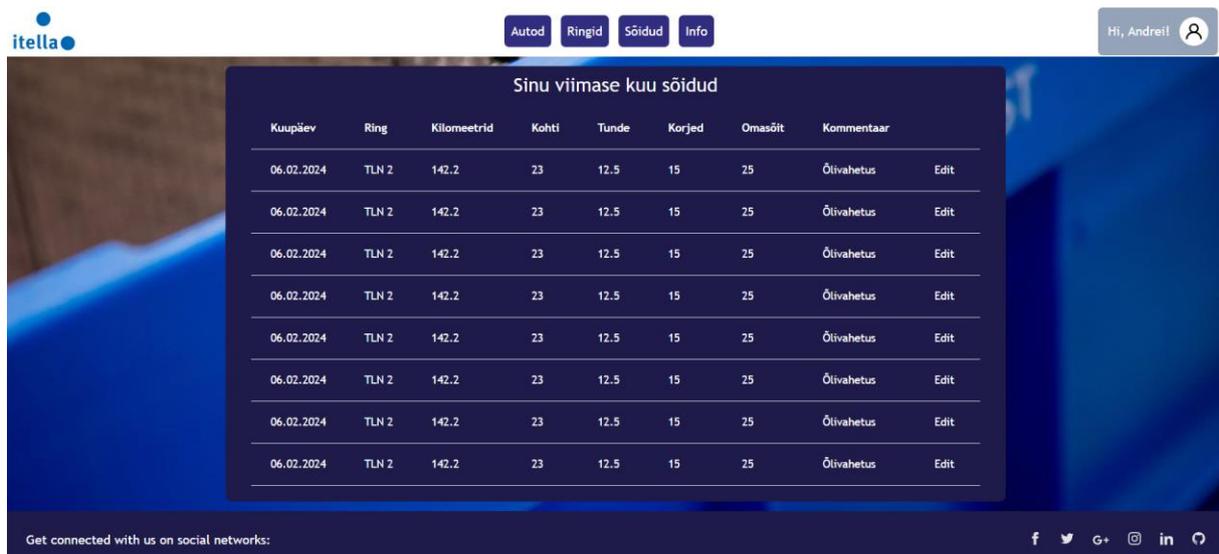
Storing username and password

As soon as it comes to identity data, it's concluded that storing username and password as a plain text is a horrible idea, that's why it is a must to store not the passwords itself, but hash values of these passwords. In other words, when the user creates the password, algorithm throws it away after hashing and stores only hashed value. After successful registration, the user inserts the same password and if the hash of it is the same as it is in the database, the user typed in correct password. So, the author needs to provide correct one-way algorithm to ensure that courier's personal data would not be interfered with.

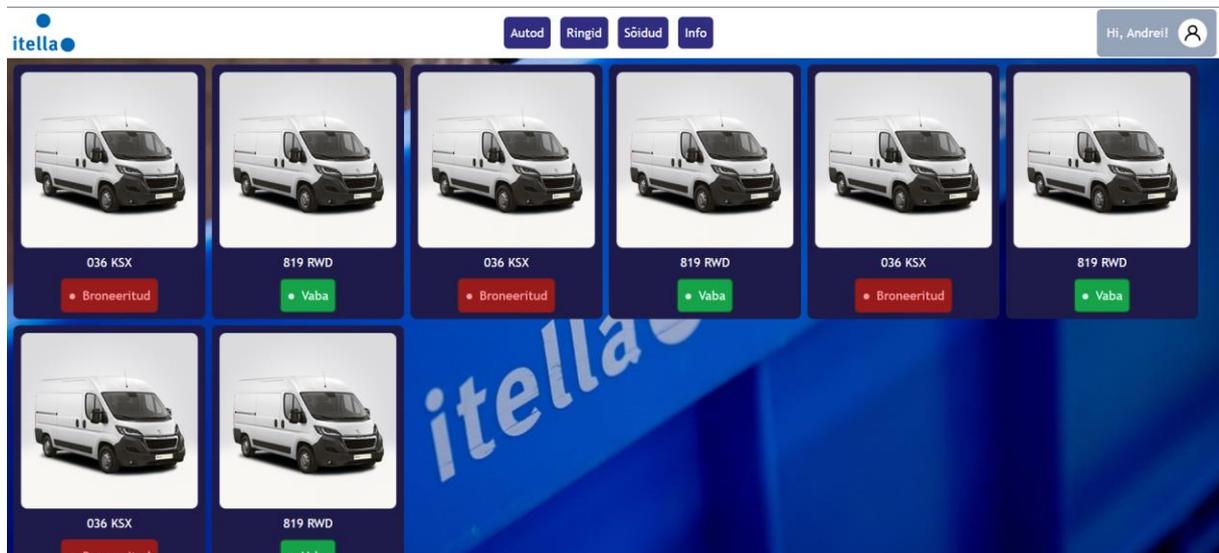
4.5 User Interface

As was already mentioned, the user interface must be minimalistic and intuitive and provide all necessary info about work the courier needs to do. Author came up with the idea of a minimalistic navigation bar, main section and a footer.

NB! The pictures below demonstrate only demo vision of the application. It will change as author will proceed working on it and end result may vary.



Pic. 2. Main page if Itella Web Application



Pic. 3. Car view of the page

4.6 Project implementation

In author's project, he will use the repository pattern to separate business and database logic, because its more flexible and allows to reuse code in any upcoming projects that may come. Also, the API controller is created to manipulate incoming data from open-source API services. The main database generation script based on the ERD schema above is provided below in the "Extras" section.

5 Summary

6 Sources used

1. Password hashing - <https://stackoverflow.com/questions/1054022/best-way-to-store-password-in-database> – (25.02.2024)
2. Is it reasonable to store API data in DB? - <https://softwareengineering.stackexchange.com/questions/442626/is-it-a-good-idea-to-design-a-database-based-on-the-api-response-data-types#:~:text=Depends%20on%20the%20size%20of,changes%20to%20your%20data%20base%20schema> – (25.02.2024)
3. Pulled API in Database - <https://stackoverflow.com/questions/73375674/should-pulled-api-data-be-stored-inside-a-database> – (25.02.2024)
4. Image storing in Database - https://www.reddit.com/r/Database/comments/rx38ip/database_for_storing_images/ - (25.02.2024)

7 Extras

Database script (attribute names may vary depending on database engine)

```
CREATE TABLE address (  
    Id integer NOT NULL,  
    Country varchar(25) NOT NULL,  
    City varchar(50) NOT NULL,  
    PostalCode varchar(10) NOT NULL,  
    Street varchar(50) NOT NULL,  
    HouseNr char(10) NOT NULL,  
    ApartNr char(3) NULL,  
    CONSTRAINT address_pk PRIMARY KEY (Id)  
);
```

-- Table: auto

```
CREATE TABLE auto (  
    Id integer NOT NULL,  
    CarRegNr varchar(8) NOT NULL,  
    CarStatus smallint NOT NULL,  
    Mileage integer NOT NULL,  
    CONSTRAINT auto_pk PRIMARY KEY (Id)  
);
```

-- Table: contact

```
CREATE TABLE contact (  
    Id integer NOT NULL,  
    UserId integer NOT NULL,  
    PhoneCodeId integer NOT NULL,  
    PhoneNumber integer NOT NULL,  
    AddressId integer NULL,  
    CONSTRAINT contact_pk PRIMARY KEY (Id)  
);
```

-- Table: locker

```
CREATE TABLE locker (  
    Id integer NOT NULL,
```

```

RoundId integer NOT NULL,
AddressId integer NOT NULL,
LockerName varchar(40) NOT NULL,
CONSTRAINT locker_pk PRIMARY KEY (Id)
);

-- Table: message
CREATE TABLE message (
  Id integer NOT NULL,
  UserId integer NOT NULL,
  ReceptientUserId integer NOT NULL,
  ParentMessageId integer NOT NULL,
  Text varchar(2500) NOT NULL,
  Filename varchar(150) NULL,
  CreatedAt timestamp NOT NULL,
  IsRead smallint NOT NULL,
  CONSTRAINT message_pk PRIMARY KEY (Id)
);

-- Table: phone_code
CREATE TABLE phone_code (
  Id integer NOT NULL,
  Code varchar(4) NOT NULL,
  Country char(2) NOT NULL,
  CONSTRAINT phone_code_pk PRIMARY KEY (Id)
);

-- Table: role
CREATE TABLE role (
  Id integer NOT NULL,
  RoleName varchar(50) NOT NULL,
  CONSTRAINT role_pk PRIMARY KEY (Id)
);

-- Table: round

```

```
CREATE TABLE round (  
  Id integer NOT NULL,  
  RoundName varchar(15) NOT NULL,  
  EstimatedTime float(4) NOT NULL,  
  CONSTRAINT round_pk PRIMARY KEY (Id)  
);
```

-- Table: trip

```
CREATE TABLE trip (  
  Id integer NOT NULL,  
  UserId integer NOT NULL,  
  AutoId integer NOT NULL,  
  RoundId integer NOT NULL,  
  TripStartAt timestamp NOT NULL,  
  TripEndAt timestamp NOT NULL,  
  "Date" date NOT NULL,  
  PlaceCount integer NOT NULL,  
  TakeawayCount integer NULL,  
  SelfDrive integer NULL,  
  "Comment" varchar(500) NULL,  
  CONSTRAINT trip_pk PRIMARY KEY (Id)  
);
```

-- Table: user

```
CREATE TABLE "user" (  
  Id integer NOT NULL,  
  RoleId integer NOT NULL,  
  Name varchar(50) NOT NULL,  
  Surname varchar(50) NOT NULL,  
  UserName varchar(15) NOT NULL,  
  Password varchar(255) NULL,  
  CONSTRAINT user_pk PRIMARY KEY (Id)  
);
```

-- foreign keys

```

-- Reference: contact_address (table: contact)
ALTER TABLE contact ADD CONSTRAINT contact_address
    FOREIGN KEY (AddressId)
    REFERENCES address (Id);

-- Reference: contact_phone_code (table: contact)
ALTER TABLE contact ADD CONSTRAINT contact_phone_code
    FOREIGN KEY (PhoneCodeId)
    REFERENCES phone_code (Id);

-- Reference: contact_user (table: contact)
ALTER TABLE contact ADD CONSTRAINT contact_user
    FOREIGN KEY (UserId)
    REFERENCES "user" (Id);

-- Reference: locker_address (table: locker)
ALTER TABLE locker ADD CONSTRAINT locker_address
    FOREIGN KEY (AddressId)
    REFERENCES address (Id);

-- Reference: locker_round (table: locker)
ALTER TABLE locker ADD CONSTRAINT locker_round
    FOREIGN KEY (RoundId)
    REFERENCES round (Id);

-- Reference: message_message (table: message)
ALTER TABLE message ADD CONSTRAINT message_message
    FOREIGN KEY (ParentMessageId)
    REFERENCES message (Id);

-- Reference: message_user (table: message)
ALTER TABLE message ADD CONSTRAINT message_user
    FOREIGN KEY (UsrerId)
    REFERENCES "user" (Id);

```

```
-- Reference: message_user_2 (table: message)
ALTER TABLE message ADD CONSTRAINT message_user_2
    FOREIGN KEY (ReceipientUserId)
    REFERENCES "user" (Id);
```

```
-- Reference: trip_auto (table: trip)
ALTER TABLE trip ADD CONSTRAINT trip_auto
    FOREIGN KEY (AutoId)
    REFERENCES auto (Id);
```

```
-- Reference: trip_round (table: trip)
ALTER TABLE trip ADD CONSTRAINT trip_round
    FOREIGN KEY (RoundId)
    REFERENCES round (Id);
```

```
-- Reference: trip_user (table: trip)
ALTER TABLE trip ADD CONSTRAINT trip_user
    FOREIGN KEY (UserId)
    REFERENCES "user" (Id);
```

```
-- Reference: user_role (table: user)
ALTER TABLE "user" ADD CONSTRAINT user_role
    FOREIGN KEY (RoleId)
    REFERENCES role (Id);
```