Rytis Čepulis 3kr. 1gr. 1pgr.

PSP LW2 in progress

Regitra

Context

The system is going to be used only in Lithuania.

Users:

* All users are adults (people above 18 years old).
* Users can be ether employees of Regitra or customers of Regitra.
* The users will mainly be Lithuanians, however foreigners may use it occasionally as well (for example a foreigner can register to a driving exam).

Load:

* is going to be the only system for managing traffic related services in Lithuania.
* Around 1.6million potential users considering Lithuania has around 2.2million adult occupants (source <https://worldpopulationreview.com/>) and around 1.6million of them has at least one category of driving license (according to Regitra).

Dependencies to external systems:

* E-Government Gateway ([www.epaslaugos.lt](http://www.epaslaugos.lt)) – For user identity authentication.

Use Cases

Use cases are the main functions that the user should expect from the Regitra system. Users of Regitra needs a way to:

* Manage driver’s license orders, that is three main functions:
  1. Order a new driver’ license.
  2. Cancel the order.
  3. Change order details.
* Manage registration for driving exam:
  1. Register for an exam.
  2. Change registration date if already registered.
  3. Cancel registration.
* Manage registration for a vehicle:
  1. Register vechile
  2. Order registration plate

All of these Use Cases are displayed in Use Case Figure 1.

Diagram

Description automatically generated

Use Case Figure 1. Use Cases

Components

The system is going to be based on Clean Architecture pattern. The system is going to have UserController which is going to provide Get, Post, Put, Delete methods for the UI. BusinessLogic layer is going to realize methods needed for the controller. DataAccess layer will be needed to communicate with the Database. To authenticate users the system is going to use some external identity authenticator. These major components are displayed in Components Figure 1.

Diagram

Description automatically generated

Component Figure 1. Systems components

Displaying page to order driver’s license

Communication between layers to show page for ordering driver’s license is displayed in a sequence diagram (Figure for Order 1.)

Graphical user interface

Description automatically generated

Figure for Order . Get page for ordering driver’s license (Sequence diagram)

To display page for ordering a driver’s license UI is going to need some data that describes the main information about the driver’s license that can be ordered, that data is in PersonEntity object which is personal information and categories of vehicles that the user will be allowed to drive (Figure for Order 2.)

Text

Description automatically generated with low confidence

Figure for Order . PersonEntity

To authenticate the user and get their personal information UI is going to redirect user to an external identity authenticator which should provide users personal information in PersonDto object (Figure for Order 3.)

Graphical user interface, text, application

Description automatically generated

Figure for Order . PersonDto

After the user is authenticated and personal information PersonDto is received UI is going to call GET endpoint in UserController for getting driver’s license order and pass PersonDto as request body. The endpoint is going to return PersonEntity. UserController should stay lean, therefore it is going to get PersonEntity from BusinessLogic layer.

The BusinessLogic layer is calling DataAccess layer which gets and returns LicenseCategory object (Figure for Order 4.) from the database.

Graphical user interface, text, application

Description automatically generated

Figure for Order 4. LicenseCategories

Having PersonDto and LicenseCategories now BusinessLogic can map those two into UserEntity class and return it to the UI.

Posting order for driver’s license

Communication between layers to post order for driver’s license is displayed in a sequence diagram (Figure for Order 5.)

Diagram

Description automatically generated with medium confidence

Figure for Order 5. Post order for driver’s license (Sequence diagram)

When the user wants to submit and order the UI is going to call POST method in UserController’s endpoint for getting driver’s license order and pass PersonDto as a request body. The endpoint is going to return OrderEntity (Figure for Order 6.) and status code.

Text

Description automatically generated with low confidence

Figure for Order 6. OrderEntity

If the status code is correct the UI is going to use OrderEntity to display information that the driver license will contain. UserController should stay lean, therefore it is going to BusinessLayer to get OrderEntity and add order to the database.

The BusinessLogic layer is going to map PersonDto and RegitraInformation (Figure for Order 7.) into an Order class which is going to be the same as OrderEntity and pass it to DataAccess layer which is going to Insert the new order into the database and returns Order object. Finally, the Order is mapped into OrderEntity and passed to UserController which returns it to UI.

Graphical user interface, text, application

Description automatically generated

Figure for Order 7. RegitraInformation

NOTES

Prisiregistruoja

Uzsisako pazym

Clientas postina orderControlleriui

Controlleris kreipiasi I logikos layeri (paduoda personalIdentificationNumber) ar galima uzsakyti

Logikos layeris kreipiasi I duombazes gateway u

Duombazes gatewayus gauna visas LicenseCategories kurias turi Person

Jeigu is LicenseCatogories yra zmogaus norimas uzsakyti

True ->

False ->

Kokio dydzio sistemas, vartotj skaicius, kas tie vartotojai vartotojai

diagrama kaip isskaidom servisus, komponenetus, modulius

galima pasirinkti viena dali kad per daug neissiplesti (pvz automobl. registracija, vair, pazm isdavimas)

TODO

Business logic change name into Application

Make DB and DBAcess into one in communication diagrams