Faculty of Computers, Informatics and Microelectronics
Technical University of Moldova

Web Technologies

Laboratory Work

Author:
Daniela Cojocari

Supervisor:
Tudor Plugaru

May 11, 2018

Tasks

Laboratory Tasks:

1. Understand what is an ORM and how to use it;
2. Get more familiar with MVC pattern;
3. Understand BusinessLogic pattern;
4. Understand how authentication and authorization works.

Project description

A web site for implementation inside the truck companies for management and better organisation of the work.

Each user will add, edit, and remove truck. Each added truck will be displayed for entire company, and everyone can edit or remove it, same as the user that added it. After each editing action on a posted truck, there will be added the time and date of the edit. So it will be visible most important details regarding the truck and its dispatcher.

I. Implementation

This project was implemented in asp.net, based on bundles. There are 2 modules: truck and user, 3 controllers: home (initial page), authentication (for the authentication process), user (for crud action on the trucks).

For authentication, was used module user which has following 2 required fields: e-mail, password and 2 non-required fields: name, country.

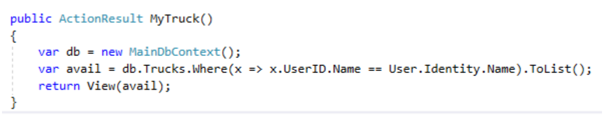
To store data, was implemented database on LocalDb. Its common to connect through MS SQL Server and manipulate the databases through migrations.

To add and use data from database was used Get-Post method. Next was added the Credential Validation and the Error Message/Adding URL Validation. With URL Validation the e-mail and password is saved.

To enable database,was connected the Database to the Application through Connection Strings.

For the password was used encryption and decryption methods.

Next step was truck table implementation, which was based on truck list module with next required fields: dispatcher's ext, truck's identity number, truck's current location and truck's current status, post time and date and edit time and date. And one of the most important field is user information of type User. With it was possible to display truck table on personal added trucks, using identity.name.



```
public ActionResult MyTruck()
{
    var db = new MainDbContext();
    var avail = db.Trucks.Where(x => x.UserID.Name == User.Identity.Name).ToList();
    return View(avail);
}
```

Figure 1: Filter for user view

II. Project view

In figure Nr.2 and Nr.3 are the main page: with a header, body and a footer

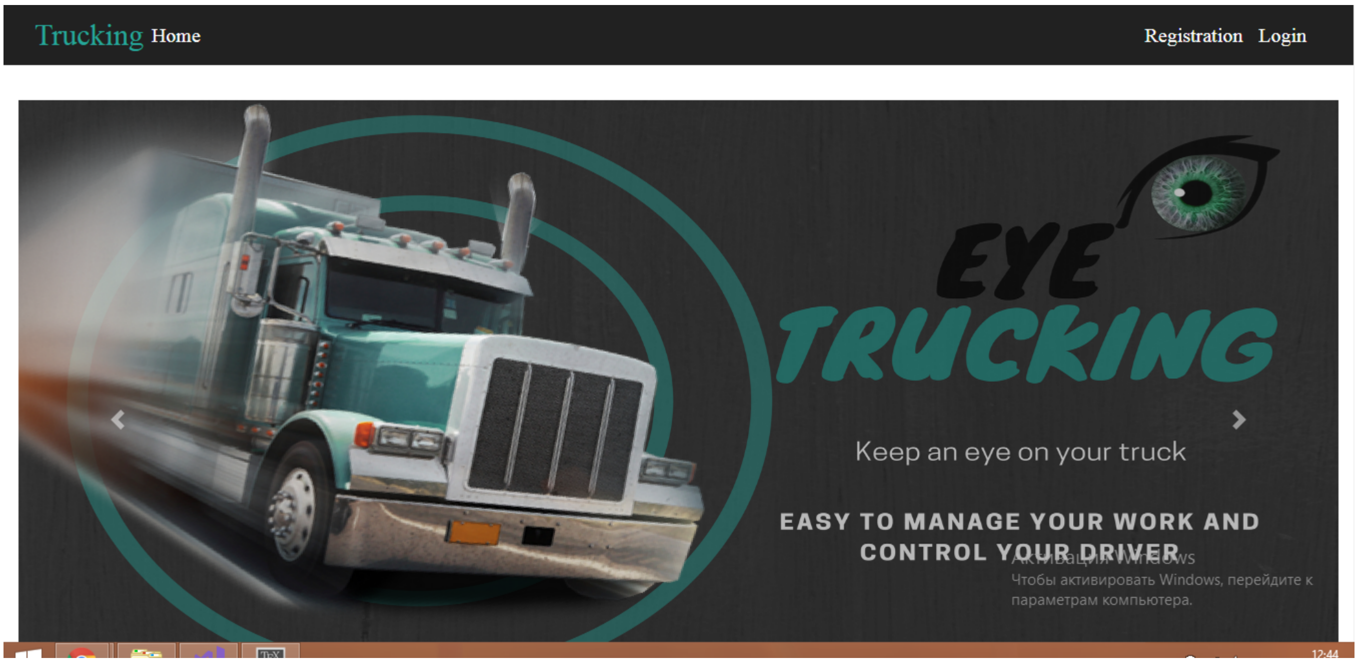
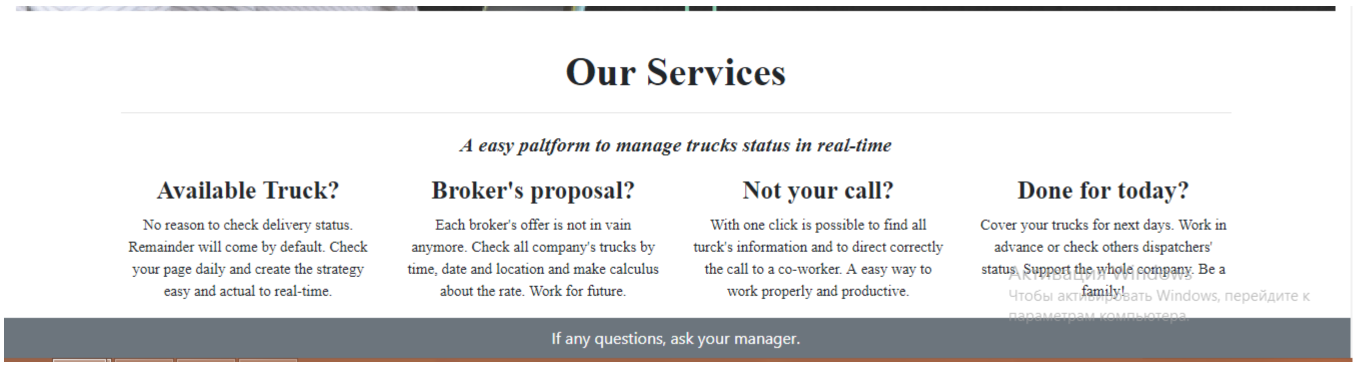

Trucking Home
Registration Login
EYE TRUCKING
Keep an eye on your truck
EASY TO MANAGE YOUR WORK AND CONTROL YOUR DRIVER

Figure 2: Initial Page



Our Services

A easy paltform to manage trucks status in real-time

Available Truck?

No reason to check delivery status. Remainder will come by default. Check your page daily and create the strategy easy and actual to real-time.

Broker's proposal?

Each broker's offer is not in vain anymore. Check all company's trucks by time, date and location and make calculus about the rate. Work for future.

Not your call?

With one click is possible to find all turck's information and to direct correctly the call to a co-worker. A easy way to work properly and productive.

Done for today?

Cover your trucks for next days. Work in advance or check others dispatchers' status. Support the whole company. Be a family!

If any questions, ask your manager.

Figure 3: Initial Page

To log in, user should be registered before. So, he should complete the register form:

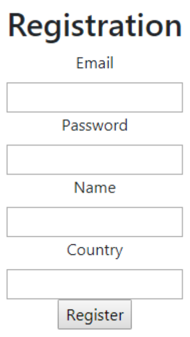


Figure 4: Registration form

And after that to login:

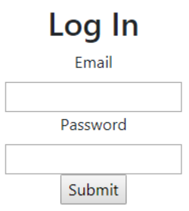


Figure 5: Login form

After log in is done, the header will change, is used same layout using a condition (@if (@User.Identity.IsAuthenticated)):

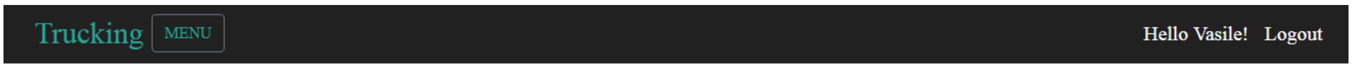


Figure 6: User authenticated - header

On Logout, user can logout himself, and return to initial page. On Menu button user can user wrapper menu:

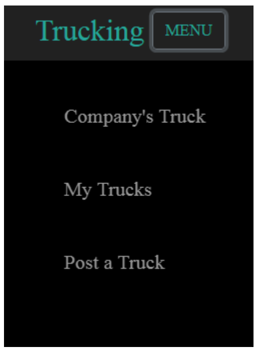


Figure 7: Menu

On Post a Truck option, view will change and user will have to complete next fields:

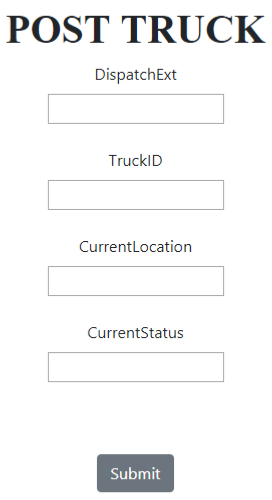
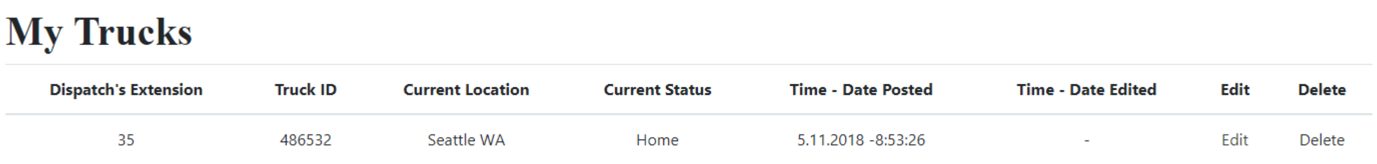


Figure 8: Post Truck form

After posting it, user will be automatically redirected on the page with his personal posts:

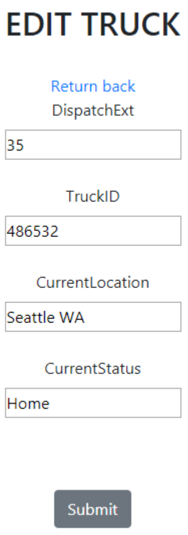


My Trucks

Dispatch's Extension	Truck ID	Current Location	Current Status	Time - Date Posted	Time - Date Edited	Edit	Delete
35	486532	Seattle WA	Home	5.11.2018 -8:53:26	-	Edit	Delete

Figure 9: My Trucks page

Is possible to edit and to delete truck. On edit option, truck is found by ID using next code line: href="@Url.Action("EditPage", "User")/@Html.DisplayFor(m =¿ item.ID)"
Also ID is used in controller:
var db = new MainDbContext();
var model1 = new Truck();
model1 = db.Trucks.Find(id);
And the editing form is opened:


EDIT TRUCK

Return back

DispatchExt	35
TruckID	486532
CurrentLocation	Seattle WA
CurrentStatus	Home

Figure 10: Edit Truck

So user can discard his action or he can enter the changes.
In case of pressing on Delete, will appear a pop-up window:

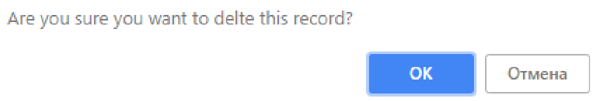

Are you sure you want to delte this record?

Figure 11: Delete Truck

If user chooses: OK option on the pop-up window, the truck will be delete:

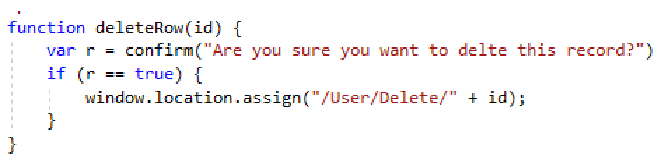
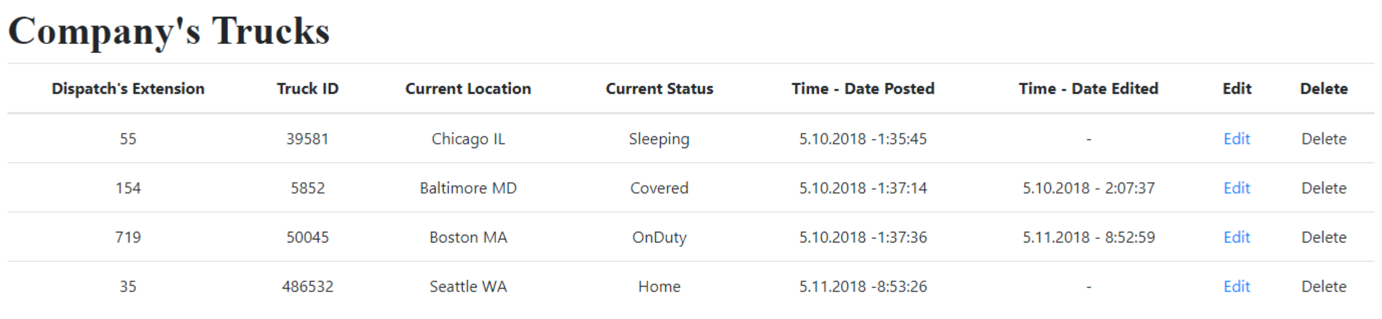

```
function deleteRow(id) {
    var r = confirm("Are you sure you want to delte this record?")
    if (r == true) {
        window.location.assign("/User/Delete/" + id);
    }
}
```

Figure 12: Pop-up Window

Truck is deleted using next code fragment:
var db = new MainDbContext();
var model = db.Trucks.Find(id);
db.Trucks.Remove(model);

To view all company's truck is used one view to display all trucks from the database:



Company's Trucks

Dispatch's Extension	Truck ID	Current Location	Current Status	Time - Date Posted	Time - Date Edited	Edit	Delete
55	39581	Chicago IL	Sleeping	5.10.2018 -1:35:45	-	Edit	Delete
154	5852	Baltimore MD	Covered	5.10.2018 -1:37:14	5.10.2018 - 2:07:37	Edit	Delete
719	50045	Boston MA	OnDuty	5.10.2018 -1:37:36	5.11.2018 - 8:52:59	Edit	Delete
35	486532	Seattle WA	Home	5.11.2018 -8:53:26	-	Edit	Delete

Figure 13: Pop-up Window

III. Conclusion

In this laboratory work were implemented a web site based on asp.net with MVC pattern, 2 models, basic Create, Read, Update, Delete operations for truck model.

Was implemented a filter for authenticated user's posts. For this to happen, was implemented registration and authentication with encrypting and decrypting of the password.