# WizMail – 9 May 2017

The WizMail is a web mail platform where the users can send and receive emails. Your task is to create a web application that has back-end and front-end logic using the provided resources (**TomEE**, **MVC Framework**, **Bootstrap** and views). You are free to edit those provided resources to fit your needs.

## Data Models

Create the required **entities**. Use the appropriate **data types**.

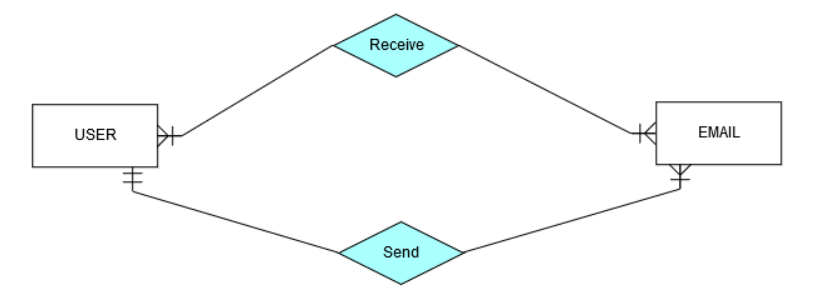
* The platform contains information about **users**, and **emails.**
* Users can **register** in the system. After successful registration, the user has **username, password, fist name and last name.**
* An **email** has **subject**, **message**, **attachment,**  **send date, sender, recipients** and **flags** wether the mail is **sent, read, draft or in trash.**

## Functionality

* **All users** can view the login page.
* **All users** can view the register page.
* **Logged-in users** can send emails to **multiple users**.
* **Logged-in users** can see received mails.
* **Logged-in users** can save and resend drafts.
* **Logged-in users** can send emails to the trash.
* **Logged-in users** can delete permanently or restore emails.
* **Guests** (anonymous users) can register an account with their own username and password
* **Guests** can login by username and password.
* **Logged-in users** can logout.
* When guest user tries to access a page that is allowed only for logged in user he should be **redirected to the login page**

## Design the Database

Design **entity classes** and create a **database** to hold the **users** and **emails**

****

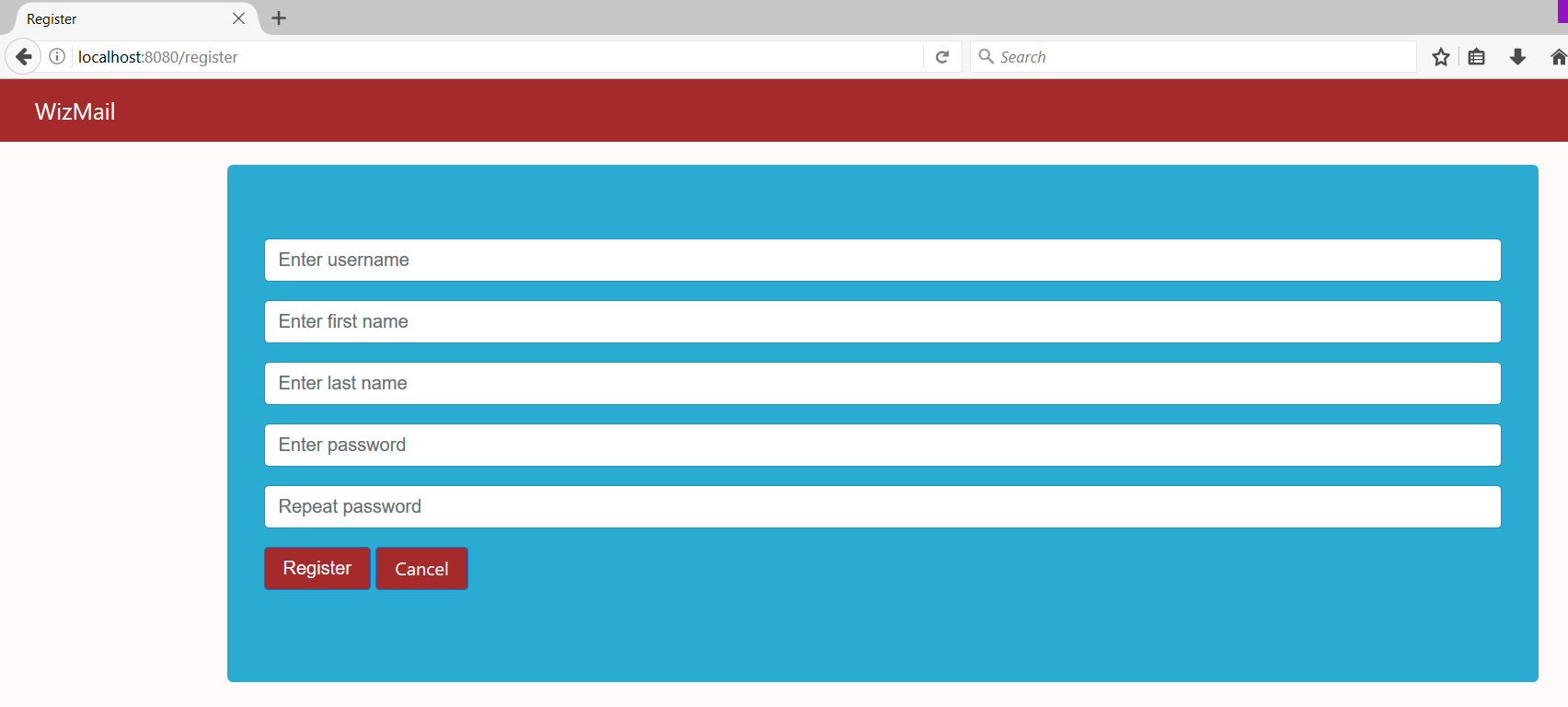
Score: **10 points**

## Implement User Registration, Login and Logout

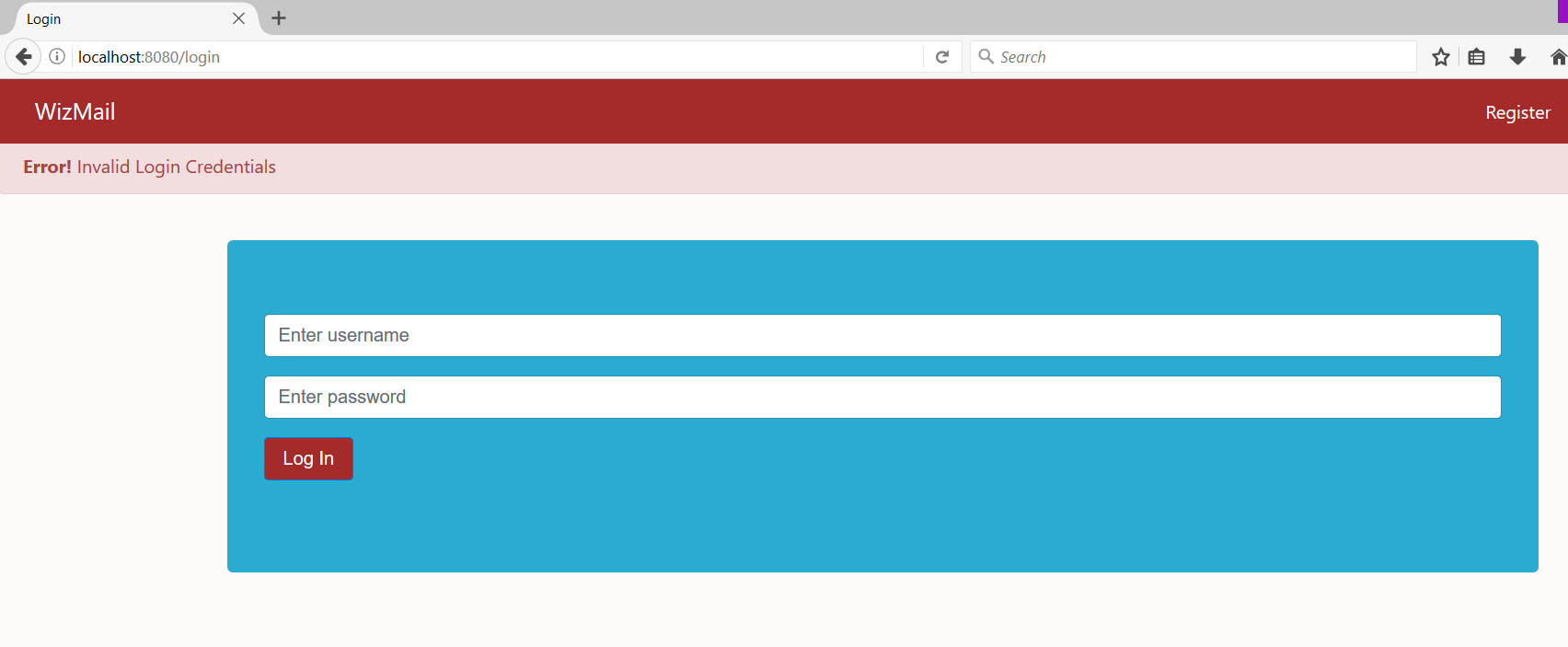
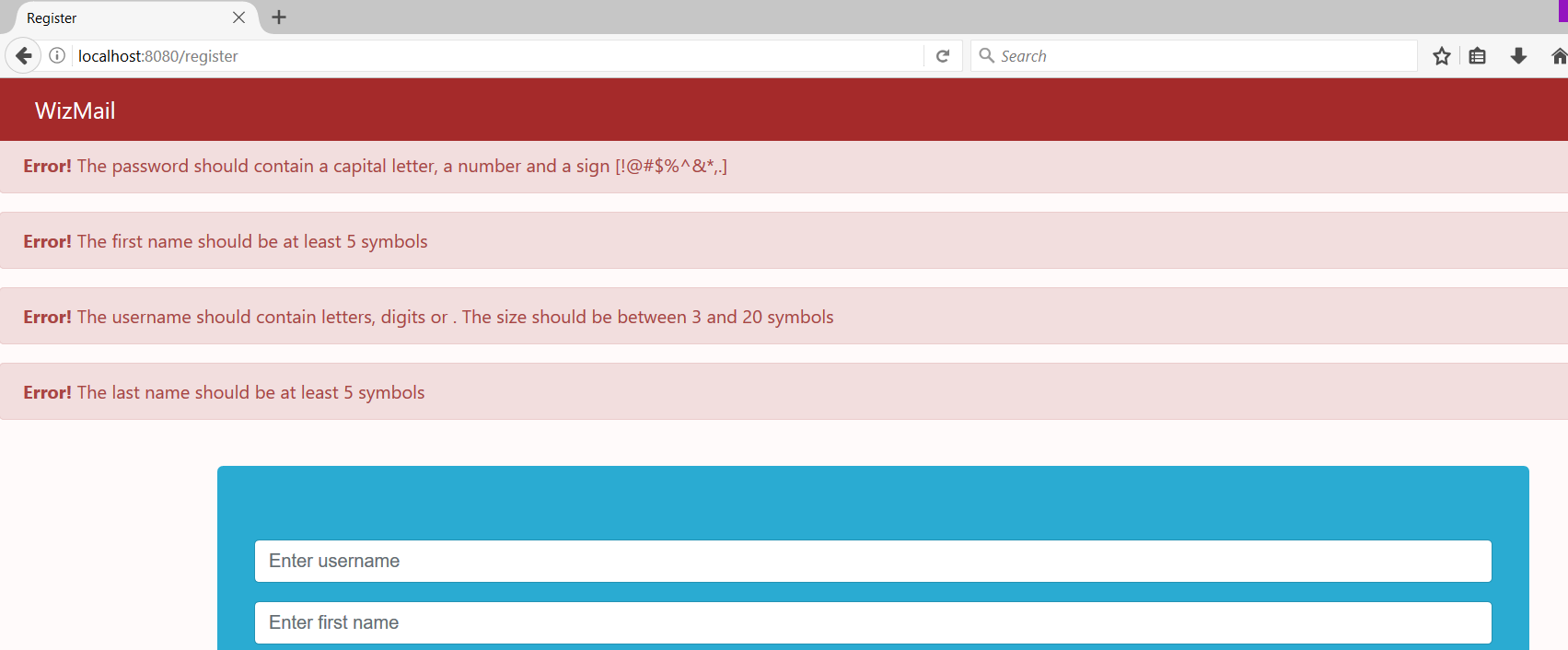
The guest users can register and log in the forum. He can provide to:

* **Register**
  + **Username** – must contain **letters, digits or “.”**. The size should be between 3 and 20 symbls. It must be unique.
  + **Password** – length must be **at least 8 symbols** and must contain at least **1 uppercase**, **1 lowercase** letter and **1 digit**
  + **Confirm Password** – must **match** the provided password
  + **First Name** – must be between 5 and 30 symbols
  + **Last Name** – must be between 5 and 30 symbols
* **Login**
  + User can log in with **username** and **password**
* **Logout**
  + When logged in the user should have option to **log out**
  + The **Logout** should redirect to the **login** page

After register, the user should be redirected to the login page. If there are any errors with the registration, they should be displayed in the register form. All validations should be performed on the back-end.



The **Cancel** button should lead to the **login page**.

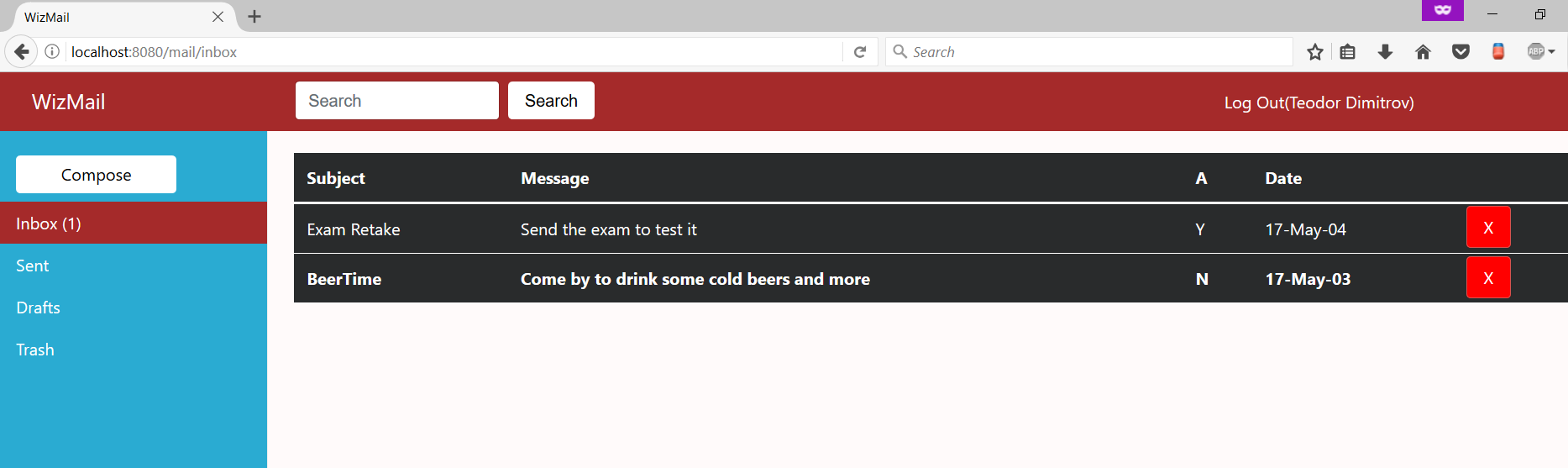


Score: **15 points**

## Implement Inbox

Implement the **Inbox** functionality. There should be all **emails** which have the **logged-in user** as a **recipient**.

* After login display **Log Out** option and show the **full name (fist name + last name)** of the user
* If the mail is with **flag read** it should **not be** bold, otherwise it should be **bold**.
* In the view limit the message up to **50 symbols**.
* **Format** the **date** as on the picture.
* If there is an attachment show **Y**, otherwise show **N**.
* **Display** the number of unread emails in **brackets** if there are any.
* **Display** the number of draft emails in **brackets** if there are any.
* If you **click** on the email go to its **details and mark the email as read**.
* If you click the **X button** send it to **trash**.

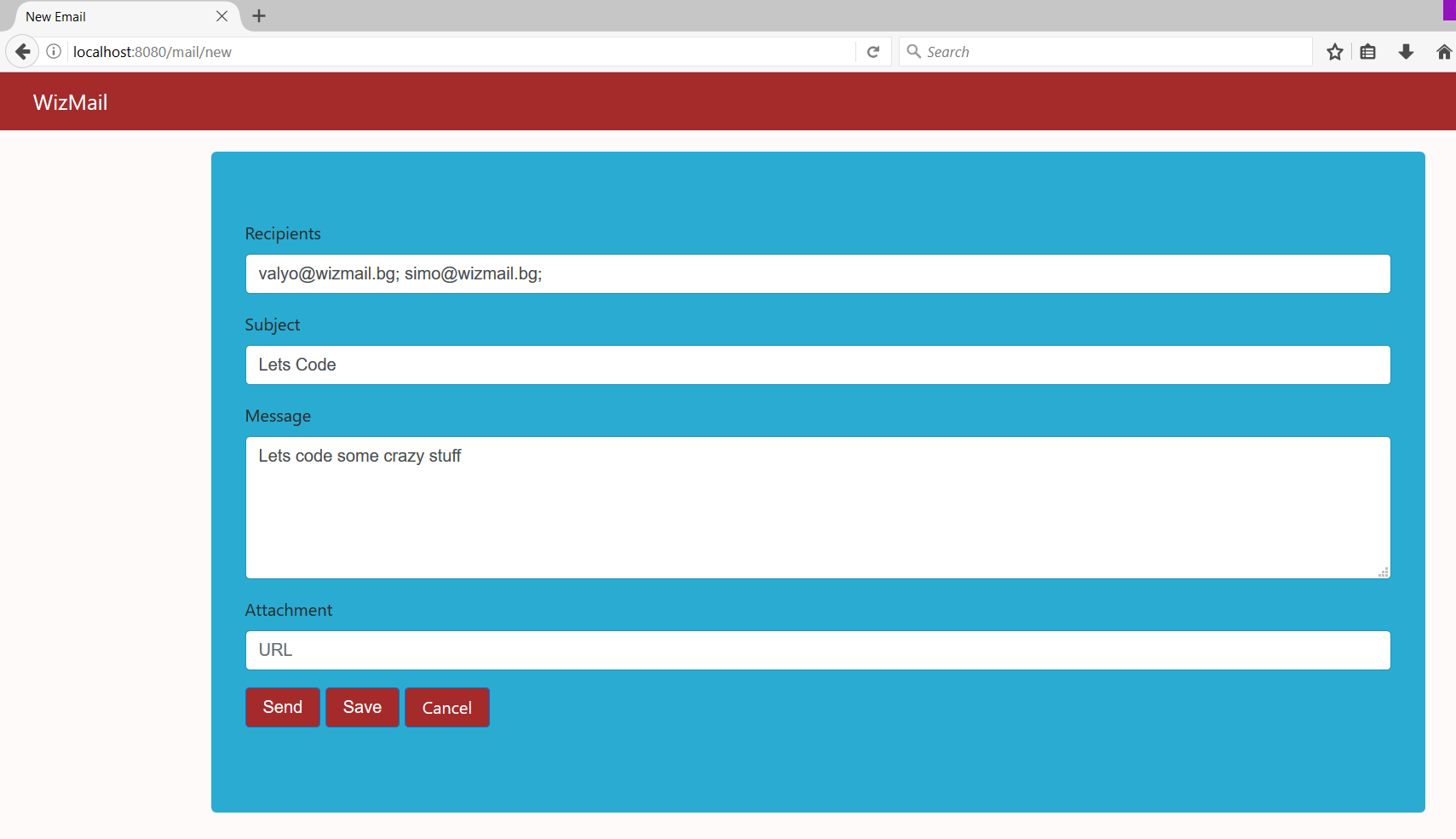


Score: **25 points**

## Implement Compose Email

When you click on **Compose** you should be able to **send** emails or **save** them as **draft**.

* The **subject** should be between **3** and **50** symbols
* The message should be up to **300** symbols
* The attachment should be up to **250** symbols
* The recipients should not be **null**
* A valid recipients list should contain the [username**@wizmail.bg**](mailto:username@wizmail.bg)**;** followed by zero or more **whitespaces**
* When **send** is clicked **create** an **email** and **send** it
* When **save** is clicked **create** an **email** and **save** it as **draft**

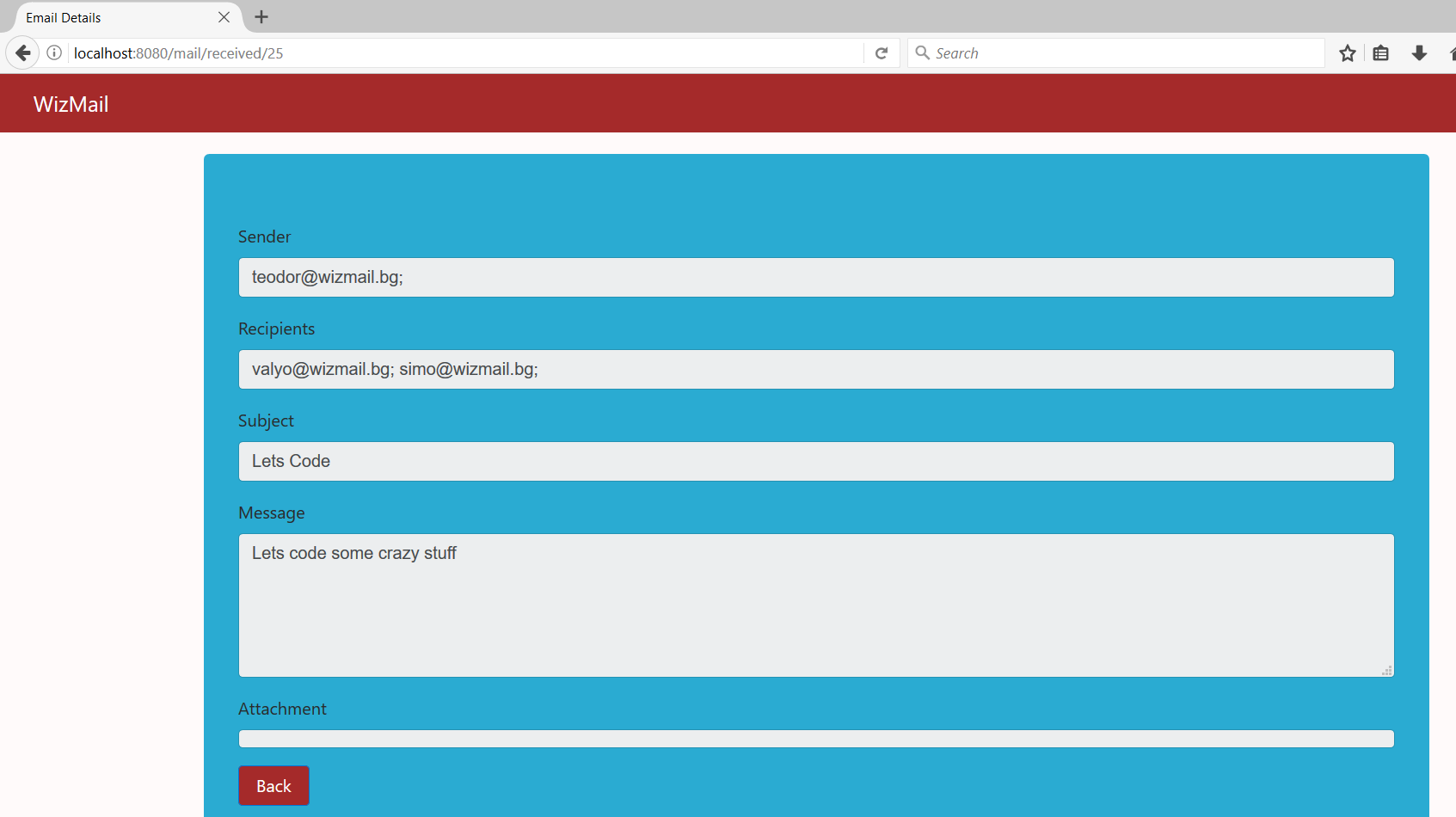
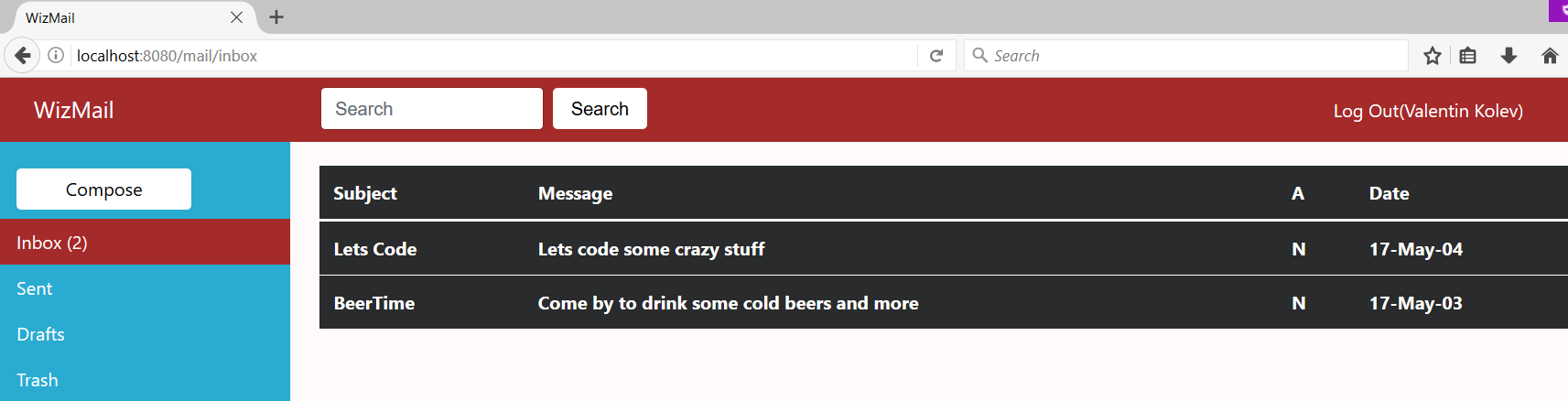


Score: **15 points**

## Implement Email Details

When an **email** is **clicked** you should see its **details**.

* The **Back** button leads to **inbox**
* Display the **details** as shown on pictures

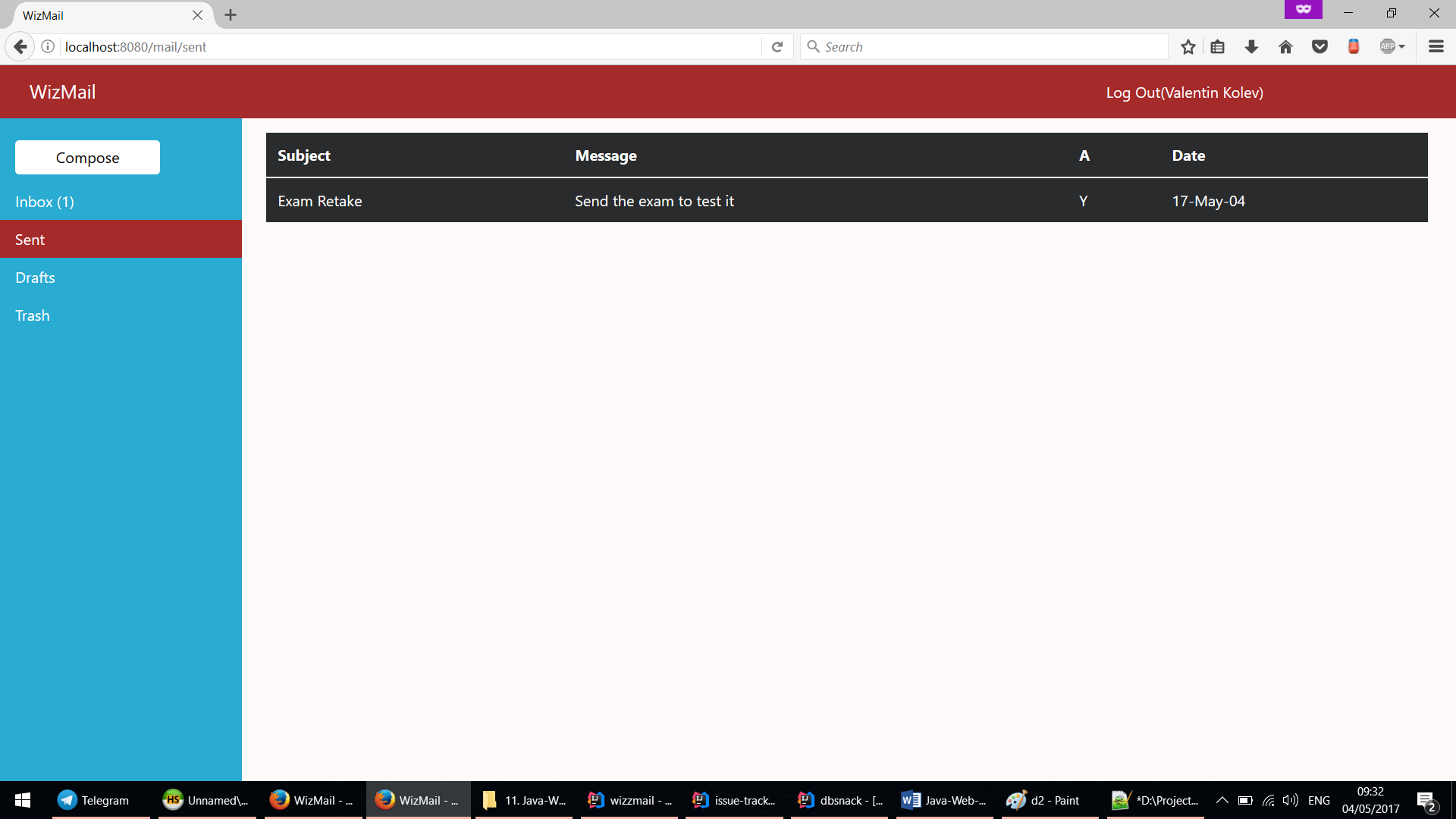


Score: **10 points**

## Implement Sent Emails

Show all sent **emails** by the **logged-in user**.

* When you click an email **don’t mark** it as **read**
* If you **click** on the email go to its **details**.
* **Display** the number of unread emails in **brackets** if there are any.
* **Display** the number of draft emails in **brackets** if there are any.

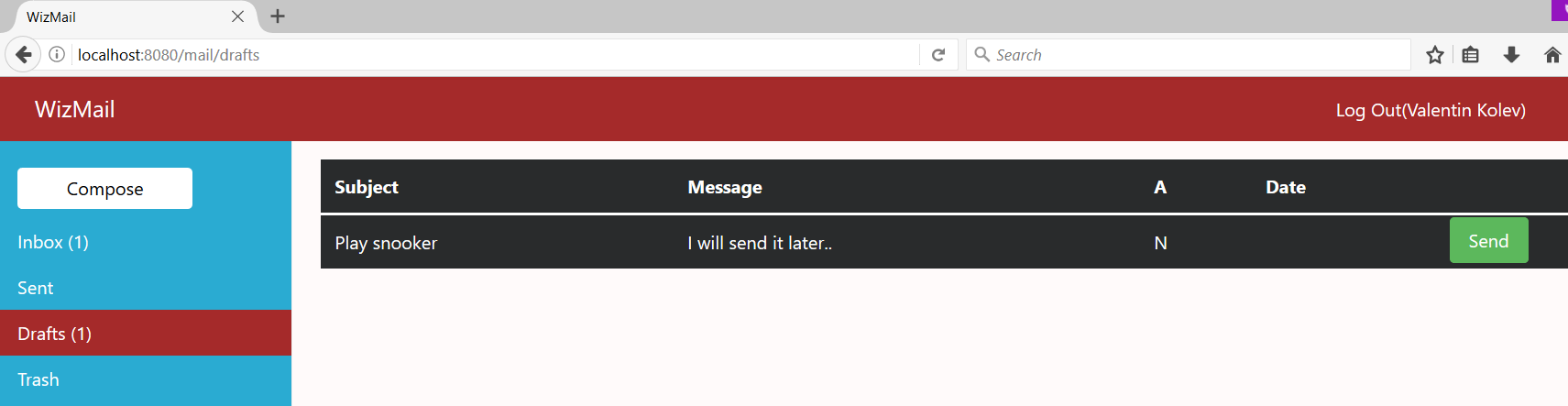


Score: **10 points**

## Implement Draft Emails

Show all emails that are **saved** from the **compose functionality.**

* When you **click** the send button **send the email**
* When you click an email **don’t mark** it as **read**
* If you **click** on the email go to its **details**.
* **Display** the number of unread emails in **brackets** if there are any.
* **Display** the number of draft emails in **brackets** if there are any.

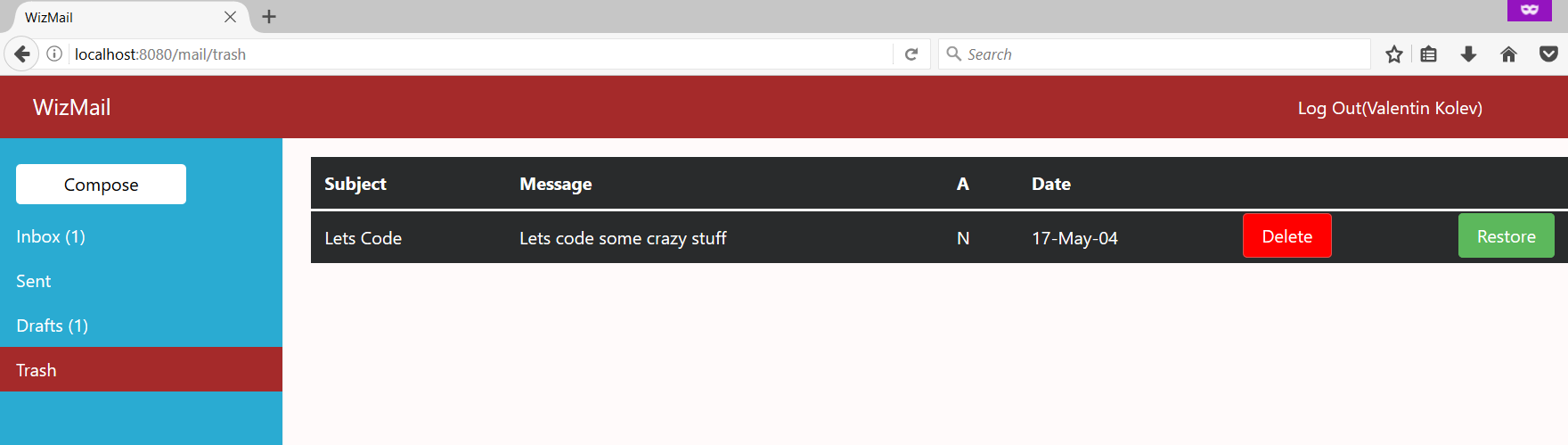


Score: **15 points**

## Bonus: Implement Trash Emails

Show all **received** mails that are **marked as trash**.

* When you click an email **mark** it as **read**
* When you **click** the **delete** button **delete the email permanently**
* When you **click** the **restore** button restore the **email back to the inbox**
* If you **click** on the email go to its **details**.
* **Display** the number of unread emails in **brackets** if there are any.
* **Display** the number of draft emails in **brackets** if there are any.



Bonus Score: **20 points**

## Bonus: Architecture

Bonus points code quality / good application structure / additional effort.

* Bonus points for implementing **separate data layer**.
* Bonus points for using **dependency inversion**.
* Bonus points for using **AutoMapper**.
* Bonus points for using **services** for different jobs.
* Bonus points for implementing **Repository** or **Unit of Work** pattern

Bonus Score: **20 points**