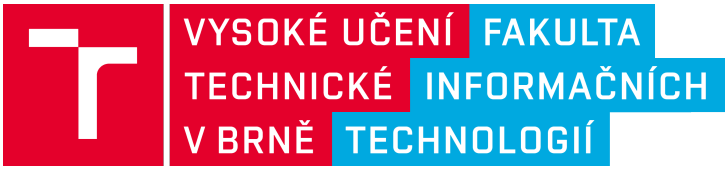


ITU
Technical report

Neonila Mashlai xmashl00

December 17, 2023

Contents

Theme

We decided to continue working on the chosen topic and make a thrift store "Garage Sale".

Changes compared to proposal

As after pitch presentation we realised, we have too little crud operations for 2 persons we had to add something in our application and we decided to add ability to chat directly in our application. It obviously has an impact on our ER-diagram so it is in additions to technical report.

Personal work

I was involved in the implementation of CRUD functionality for items.

The life cycle of an item begins on the AddItem page, which is implemented in the file AddItemPage.js. You can get to this page by clicking buttons (the Link element is imported from the react-router-dom library) on the user page, which is implemented in the UserPage.js file.

When the page is rendered, it is checked whether the user is logged in using the checkLogin() function from the Utils.js file, which checks the presence of cookies with information about the user, which are created at each login, as well as when sending a request to the server about the presence of a user with such a user id. If the user is logged in, the page is rendered and the element from the top right of the header is rendered, with the user's login, which was received through the request earlier, if not, the user is redirected to the login page via the useNavigate() hook from the react-router-dom library.

When the user is on the page, he sees a form for filling in information about the item he wants to add. Form fields contain information about the item's name, price, category, description, condition, and size. There is an element for adding an image, which also supports drag and drop. The add image element also has a validation that checks the file format that was uploaded if it doesn't match format, an error message is displayed. After filling in all the fields, the user clicks the Done button, which validates all the fields that are available mandatory, if they are not filled in, an error message is displayed. The price format is also checked to ensure that it is a number. After validation of all information, a request is sent to the server to upload the image to the hosting, which returns a reference to the image, which is then used to send a request to the server to add the item to the database. After successfully adding an item, the user is redirected to the user page, where a message about the successful addition of the item is displayed.If an error occurred while adding an item, an error message is displayed.

On the user's page, we see a list of items that have been added by the user. When you click on an item, you are redirected to the item editing page, which is implemented in the EditItemPage.js file. When the page is rendered, it is checked whether the user is logged in using the checkLogin() function from the Utils.js file. similar to the AddItem page. And also a request is sent to the server to receive information about the item that was selected for editing. If the ID of the user does not match the ID of the user who added the item, then there is redirection to the Home Page. When the user is on the page, he sees a form for editing information about the item. The form fields are already filled with information about the item, which was received through a call to the server. By default, the fields are disabled using the disabled attribute, but when you click the Edit button, they are enabled for entering information. After pressing the Enter button or exiting the field, it is automatically turned off. After entering new information, the user clicks the Done button, which validates all fields similarly to the AddItem page. If the image has been changed, first a request is sent to the server to upload the image to the hosting, similar to the AddItem page. If the image has not been changed, a request to edit the item in the database is immediately sent to the server. If an error occurred while editing an

item, an error message is displayed. After successful editing of the item, there is redirection to user page.

There is also a Delete button on the EditItemPage page, which removes the item from the database. After deleting the item, there is also redirection to user page, where a message about the successful removal of the item is displayed. .

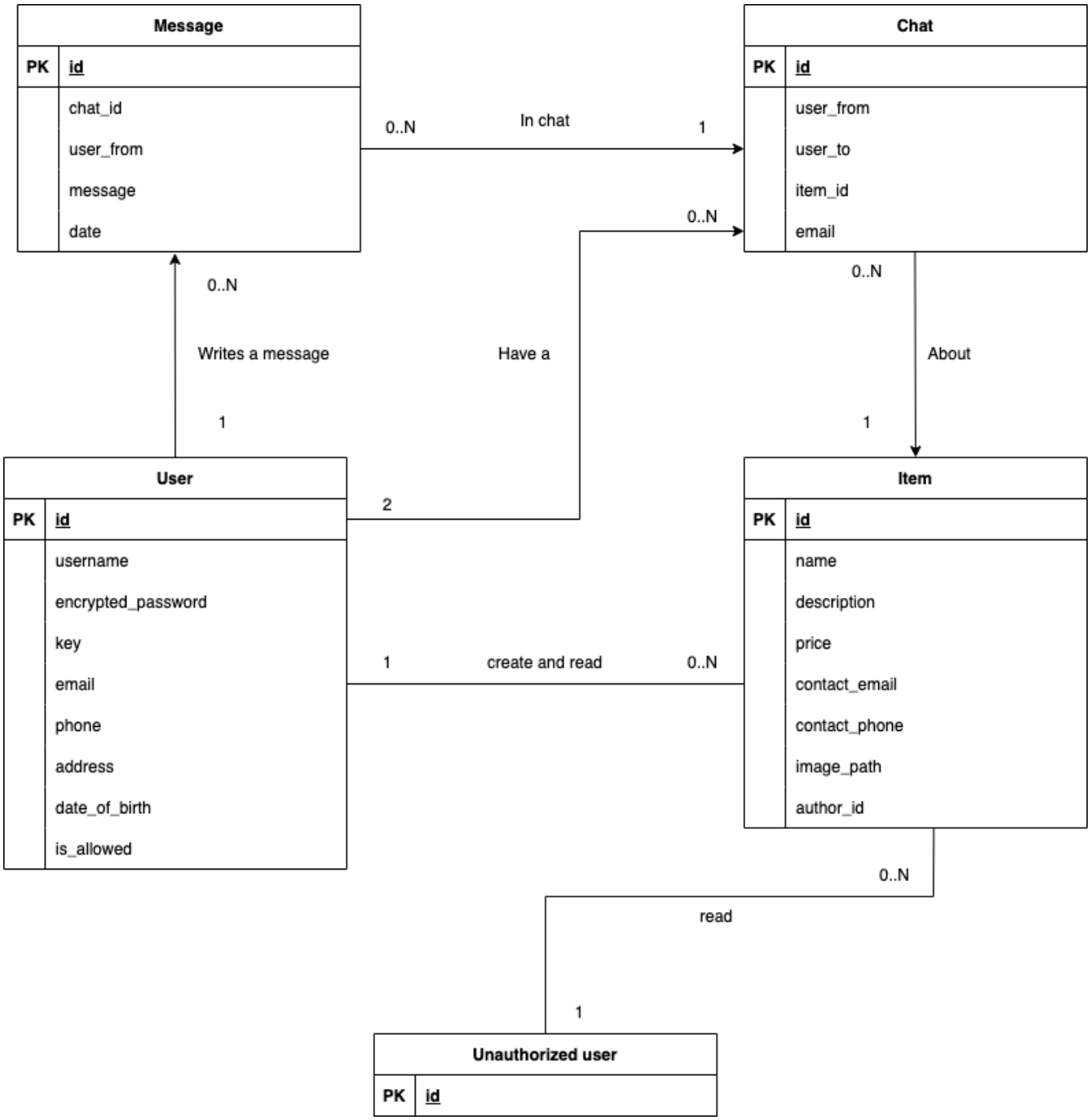
Information about which message to display on the user page is obtained through Query Params, which are passed when redirecting from the AddItemPage, EditItemPage, and DeleteItemPage pages.

An ItemsList page was also implemented in the ItemsList.js file, which displays a list of items in a certain category.

When the page is rendered, a request is sent to the server with the category parameter via the fetchItems() function. After receiving a response from the server, a page with a list of items in a certain category is rendered using the map() and addItem() functions. The addItem() function is responsible for rendering each item, which is a Link element from the react-router-dom library. Each item contains information about the name, price and image. And also in the link there are query params with the id of the item. This page is scrollable and has a footer that is always at the bottom of the page.

Also, the active global category (men, women, kids) is underlined in the header, and the local category is written on the page. When you click on an item, you are redirected to the item page, which is implemented in the ItemPage.js file, where it will be displayed.

ER-table


Message
PK
id
chat_id
user_from
message
date
Chat
PK
id
user_from
user_to
item_id
email
0..N
In chat
1
0..N
0..N
Writes a message
Have a
About
1
1
User
PK
id
username
encrypted_password
key
email
phone
address
date_of_birth
is_allowed
2
Item
PK
id
name
description
price
contact_email
contact_phone
image_path
author_id
1
create and read
0..N
0..N
read
1
Unauthorized user
PK
id