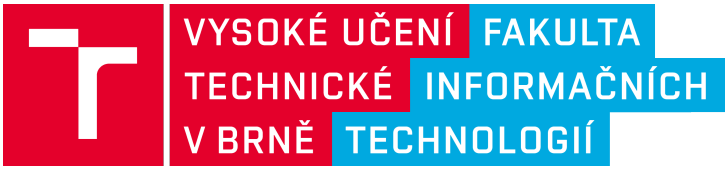


ITU
Technical report

Maksym Podhornyi xpodho08

December 17, 2023

Contents

Theme

We decided to continue working on the chosen topic and make a thrift store called "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 implementing CRUD operations for chats and messages that were added as per the feedback requirement.

The entire functionality of the chats was implemented on one page ChatsPage, which is rendered when going to /user/chats by the ChatsPage.js component.

The transition to this page occurs when clicking on the "Chats" button on the user's profile page, which is rendered by the UserPage.js component.

A new chat can be created by clicking on the "Contact seller" button on the ad page, which is rendered by the ItemPage.js component. When you click on the "Contact seller" button, a request to create a new chat is sent with the parameters id of the ad, id of the user who created the ad, id of the user who clicked on the "Contact seller" button. If the chat has already been created, the API will return the id of the chat that matches these parameters. If the chat has not been created, the API will return the id of the new chat, after which, if the request is successful, the Transition occurs to the chats page with the query parameter chatId, which is equal to the chat ID and itemId - the advertisement ID. If the request is not successful, an error message is displayed.

During rendering, the ChatsPage.js component checks for the availability of the query parameters chatId and itemId. If such parameters are available - the desired chat is opened by calling the openChat function with the parameters chatId and itemId. If there are no such parameters, no chat is opened, and the message "Select chat to start messaging" is displayed instead of messages.

A chat page consists of two components: a chat container with a list of chats and a chat container where information about the item is written - its photo and name, as well as a list of messages and a field for entering a message. The chats container is rendered by the fetchChats function, which is called when the page is rendered and every 5 seconds thereafter. This function calls the API to get a list of chats in which the user is logged in. After receiving the list of chats, the login of another user in the chat is received, as well as the name of the item being discussed. After that, the chat container is rendered with a list of chats using the map function, as well as addChat, which adds the chat to the list of chats. When clicking on the chat, the chat window is rerendered, where instead of the message "Select chat to start messaging" information about the item is displayedand a list of chat messages.When the chat is opened, the fetchMessages function is called, which calls the API for receiving messages in the chat. In the future, fetchMessages is called every second while the chat is open.

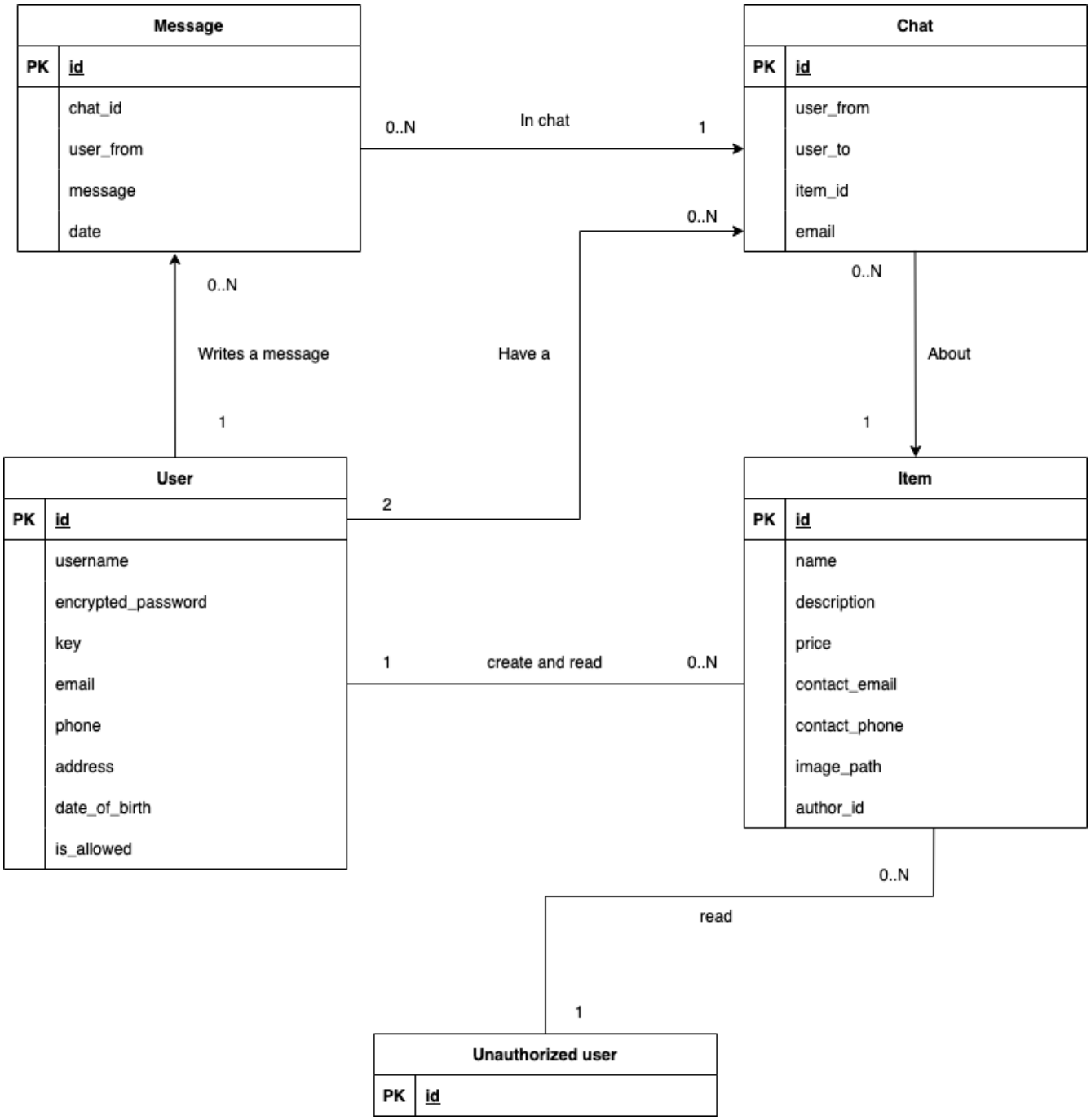
At the bottom of the chat window there is a field for entering a message and a button for sending a message. When you click on the send message button, the handleSent function is called, which sends the message to the server. If the message was successfully sent, the chat window is rerendered with a new list of messages. If the message could not be sent, an error message is displayed.

Next to each message there is a button to delete the message and to edit it. When you click on the delete button, the handleDeletemessage function is called, which requests the deletion of the message. If the request is successful, the fetchMessages function is called, which calls the API to retrieve the updated list of messages. If the request is not successful, an error message is displayed.

When you click on the edit button, the handleEditMessage function is called, which transfers the text of the message to the message input field. After the message has been edited, the handleSent function is called upon pressing the button to send the message, which makes a request to edit the message. If the message editing was successful, the fetchMessages function is called, which calls the API to get an updated list of messages. If the editing of the message failed, an error message is displayed. You can send and edit messages not only with the help of buttons, but also with the help of the Enter key.

In the chat window in the upper right corner there is a button to delete the chat. When you click on this button, the handleDeleteChat function is called, which makes a request to delete the chat. If the request is successful, the chat window is rerendered with the message "Select chat to start messaging". and the fetchChats function is called, which calls the API to get an updated list of chats. If the request is not successful, an error message is 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