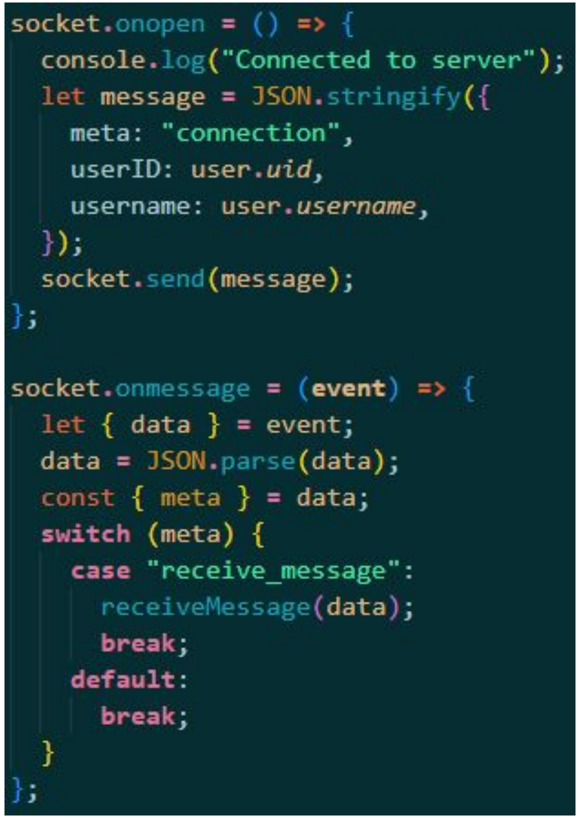
GROUPEM

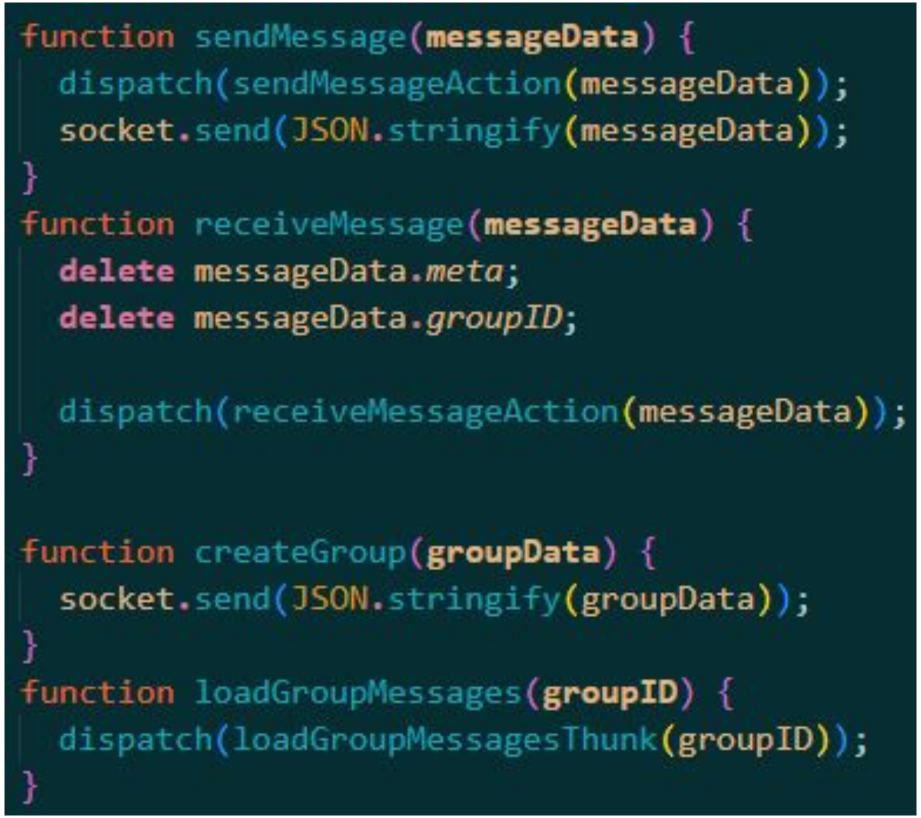
Presentació tècnica

Albert Chao Vasco
Eric Clemente Casals
Arnau Orts Brichs

useWebsocket.js

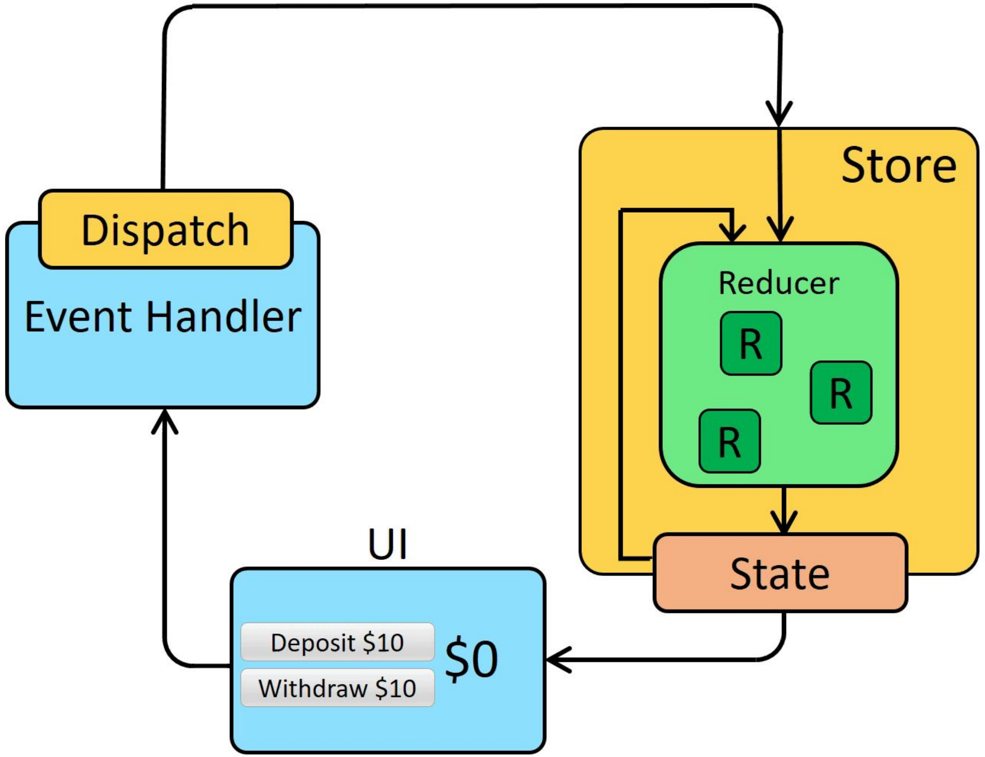


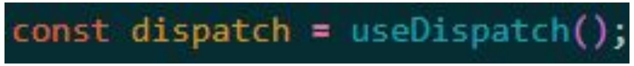
```
socket.onopen = () => {
  console.log("Connected to server");
  let message = JSON.stringify({
    meta: "connection",
    userID: user.uid,
    username: user.username,
  });
  socket.send(message);
};

socket.onmessage = (event) => {
  let { data } = event;
  data = JSON.parse(data);
  const { meta } = data;
  switch (meta) {
    case "receive_message":
      receiveMessage(data);
      break;
    default:
      break;
  }
};
```



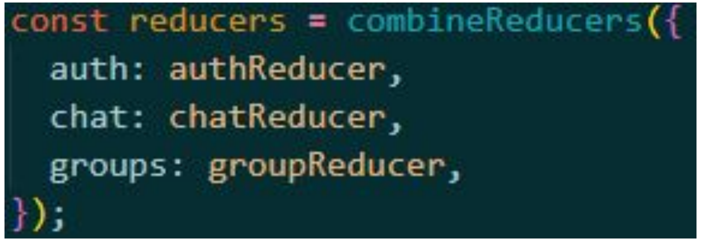
```
function sendMessage(messageData) {
  dispatch(sendMessageAction(messageData));
  socket.send(JSON.stringify(messageData));
}
function receiveMessage(messageData) {
  delete messageData.meta;
  delete messageData.groupID;

  dispatch(receiveMessageAction(messageData));
}

function createGroup(groupData) {
  socket.send(JSON.stringify(groupData));
}
function loadGroupMessages(groupID) {
  dispatch(loadGroupMessagesThunk(groupID));
}
```

REDUX

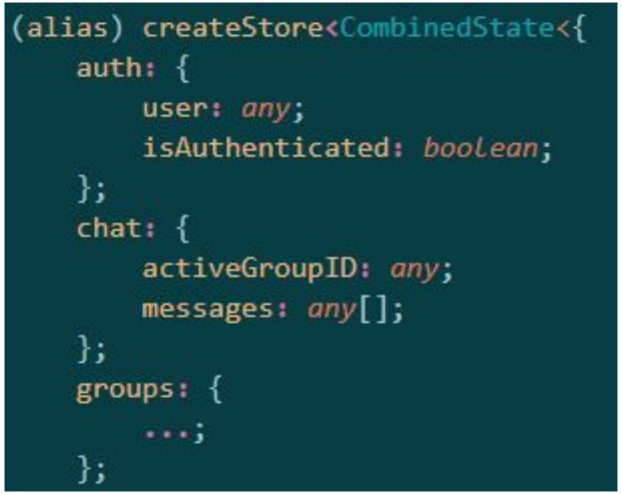

Store
Dispatch
Event Handler
Reducer
R
R
R
UI
State
Deposit $10
Withdraw $10
$0



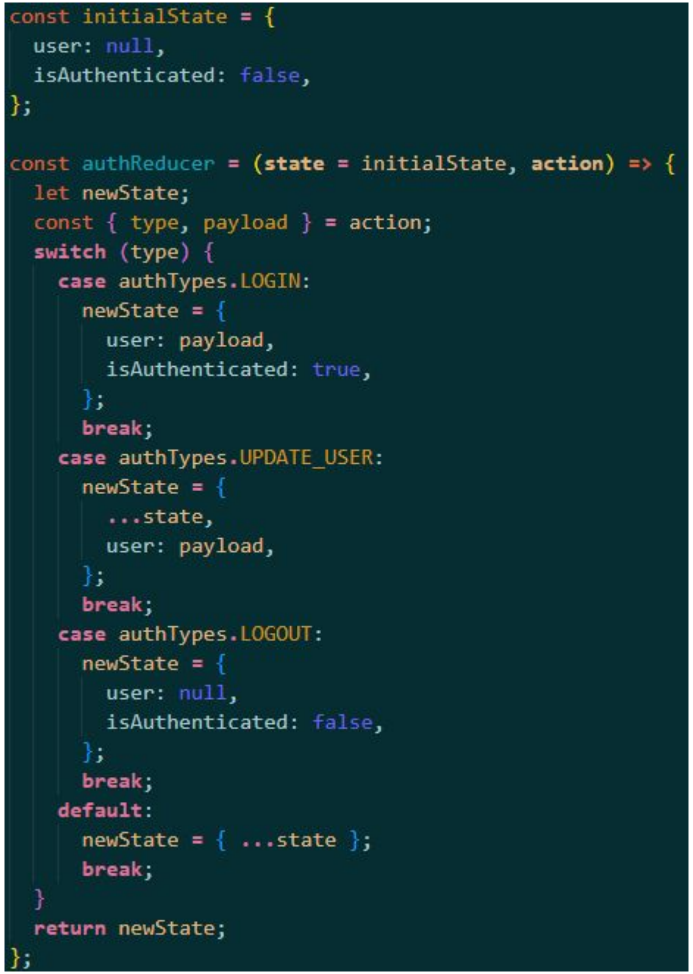
```
const dispatch = useDispatch();
```



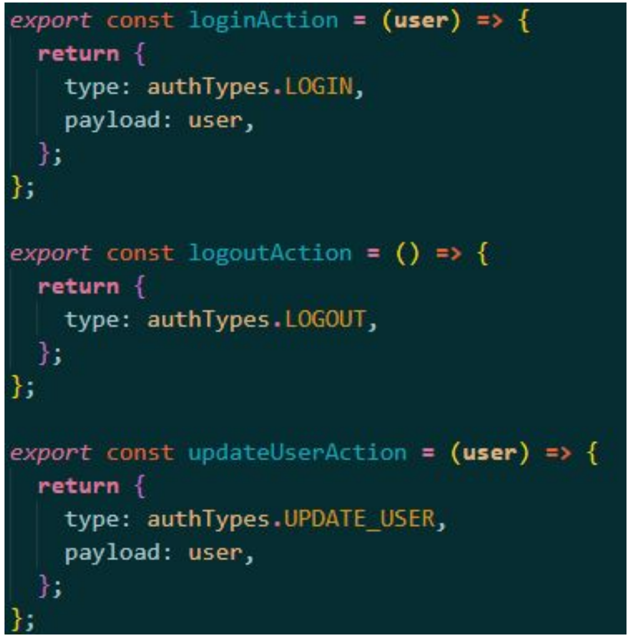
```
const reducers = combineReducers({
  auth: authReducer,
  chat: chatReducer,
  groups: groupReducer,
});
```



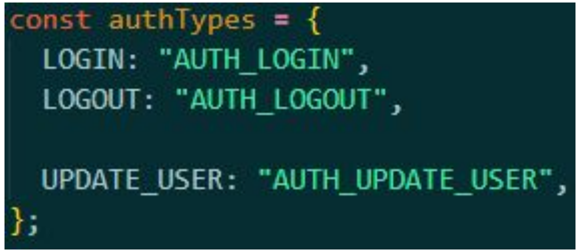
```
(alias) createStore<CombinedState<{
    auth: {
        user: any;
        isAuthenticated: boolean;
    };
    chat: {
        activeGroupID: any;
        messages: any[];
    };
    groups: {
        ...;
    };
```



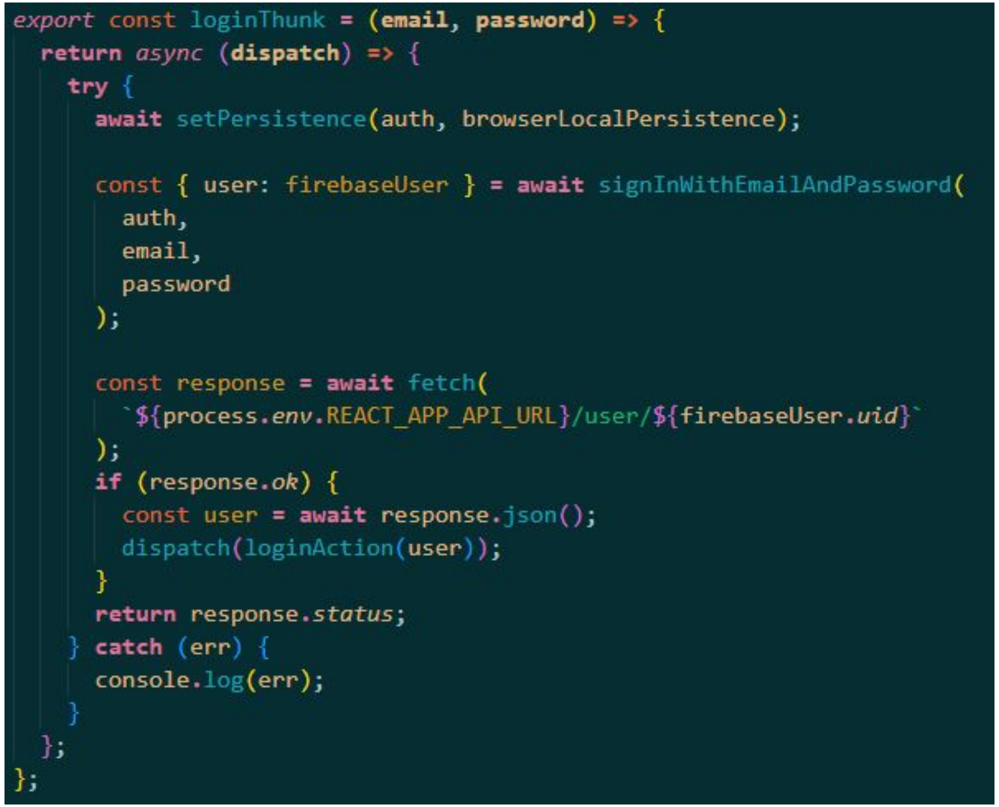
```
const initialState = {
  user: null,
  isAuthenticated: false,
};

const authReducer = (state = initialState, action) => {
  let newState;
  const { type, payload } = action;
  switch (type) {
    case authTypes.LOGIN:
      newState = {
        user: payload,
        isAuthenticated: true,
      };
      break;
    case authTypes.UPDATE_USER:
      newState = {
        ...state,
        user: payload,
      };
      break;
    case authTypes.LOGOUT:
      newState = {
        user: null,
        isAuthenticated: false,
      };
      break;
    default:
      newState = { ...state };
      break;
  }
  return newState;
};
```



```javascript
export const loginAction = (user) => {
  return {
    type: authTypes.LOGIN,
    payload: user,
  };
};

export const logoutAction = () => {
  return {
    type: authTypes.LOGOUT,
  };
};

export const updateUserAction = (user) => {
  return {
    type: authTypes.UPDATE_USER,
    payload: user,
  };
};
```

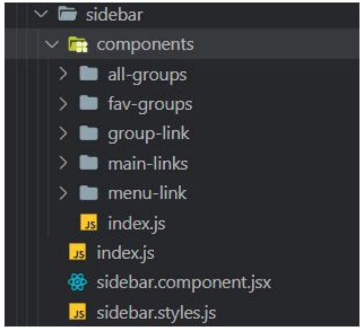


```javascript
const authTypes = {
  LOGIN: "AUTH_LOGIN",
  LOGOUT: "AUTH_LOGOUT",

  UPDATE_USER: "AUTH_UPDATE_USER",
};
```

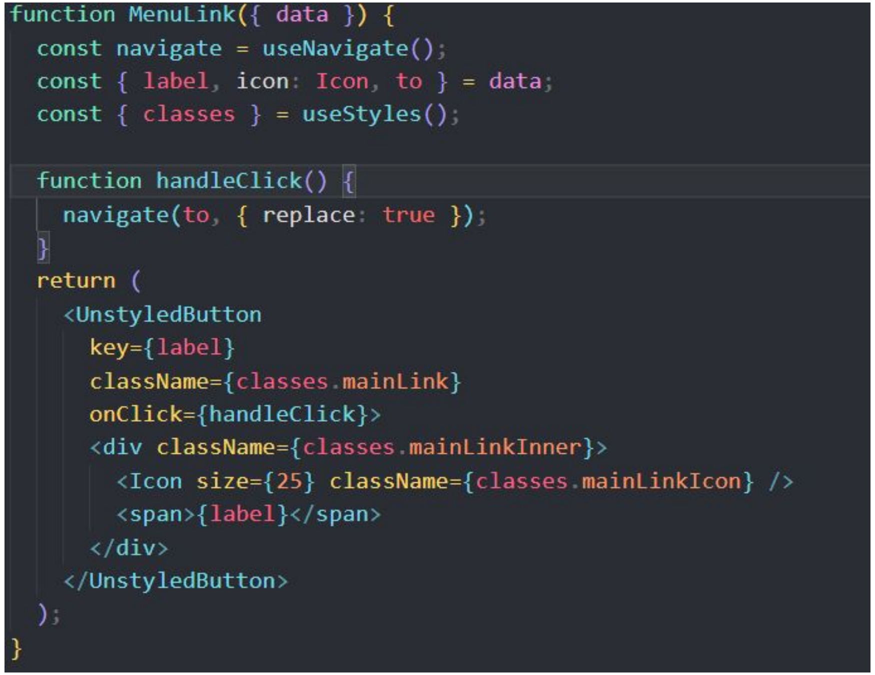


```javascript
export const loginThunk = (email, password) => {
  return async (dispatch) => {
    try {
      await setPersistence(auth, browserLocalPersistence);

      const { user: firebaseUser } = await signInWithEmailAndPassword(
        auth,
        email,
        password
      );

      const response = await fetch(
        `${process.env.REACT_APP_API_URL}/user/${firebaseUser.uid}`
      );
      if (response.ok) {
        const user = await response.json();
        dispatch(loginAction(user));
      }
      return response.status;
    } catch (err) {
      console.log(err);
    }
  };
};
```

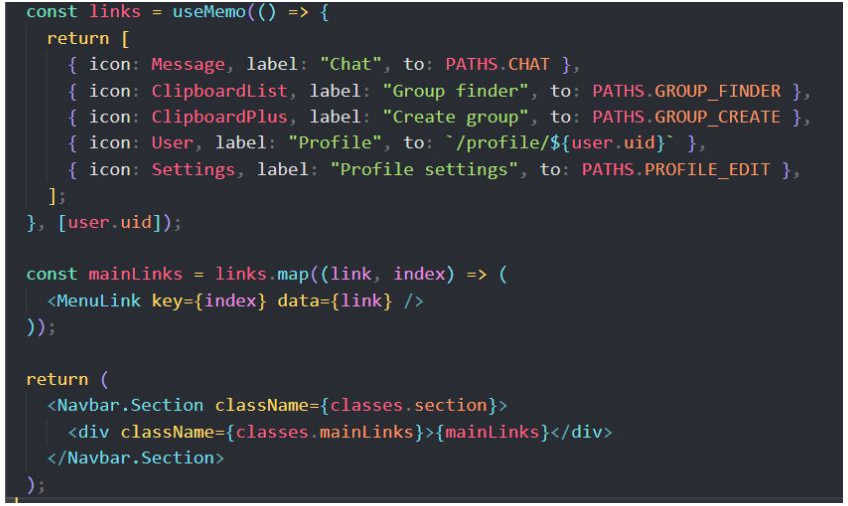
Sidebar components



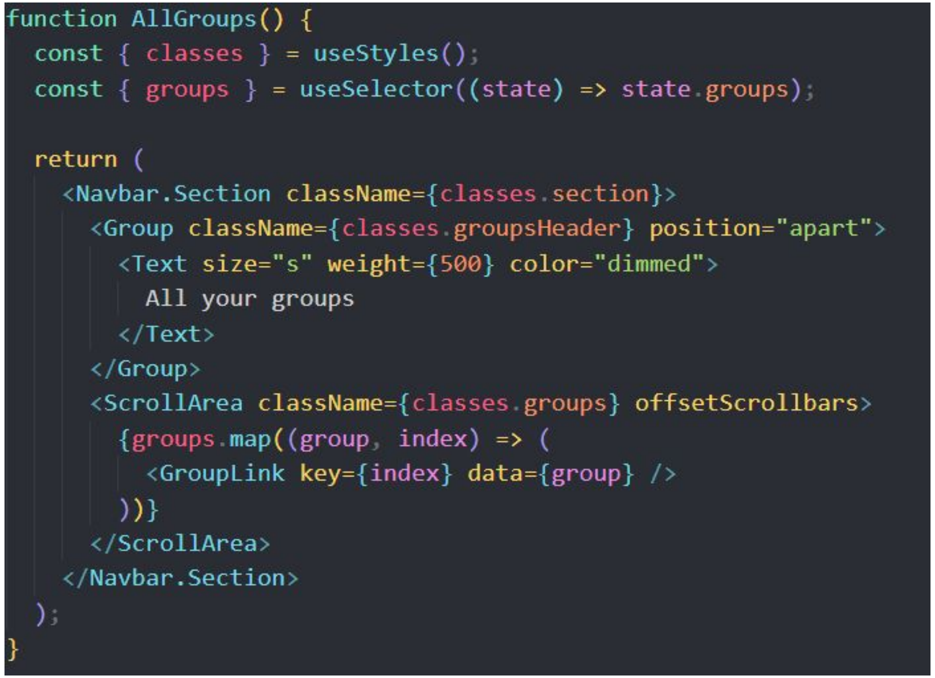
```
sidebar
  components
    > all-groups
    > fav-groups
    > group-link
    > main-links
    > menu-link
      index.js
    index.js
    sidebar.component.jsx
    sidebar.styles.js
```



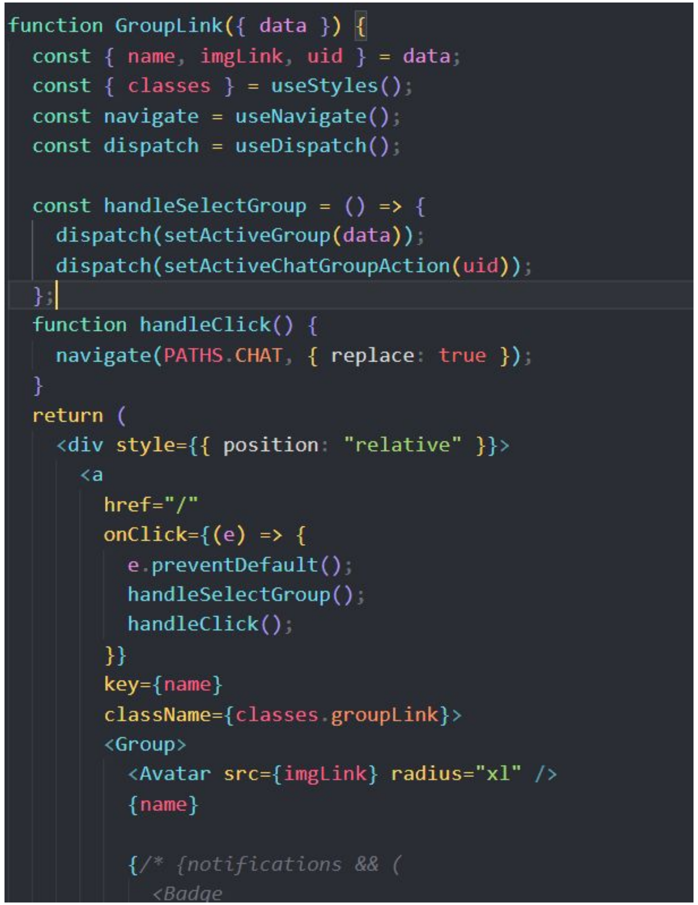
```
function MenuLink({ data }) {
  const navigate = useNavigate();
  const { label, icon: Icon, to } = data;
  const { classes } = useStyles();

  function handleClick() {
    navigate(to, { replace: true });
  }
  return (
    <UnstyledButton
      key={label}
      className={classes.mainLink}
      onClick={handleClick}>
      <div className={classes.mainLinkInner}>
        <Icon size={25} className={classes.mainLinkIcon} />
        <span>{label}</span>
      </div>
    </UnstyledButton>
  );
}
```



```
const links = useMemo(() => {
  return [
    { icon: Message, label: "Chat", to: PATHS.CHAT },
    { icon: ClipboardList, label: "Group finder", to: PATHS.GROUP_FINDER },
    { icon: ClipboardPlus, label: "Create group", to: PATHS.GROUP_CREATE },
    { icon: User, label: "Profile", to: `/profile/${user.uid}` },
    { icon: Settings, label: "Profile settings", to: PATHS.PROFILE_EDIT },
  ];
}, [user.uid]);

const mainLinks = links.map((link, index) => (
  <MenuLink key={index} data={link} />
));

return (
  <Navbar.Section className={classes.section}>
    <div className={classes.mainLinks}>{mainLinks}</div>
  </Navbar.Section>
);
```

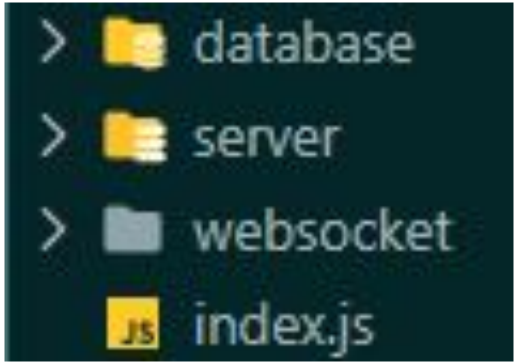


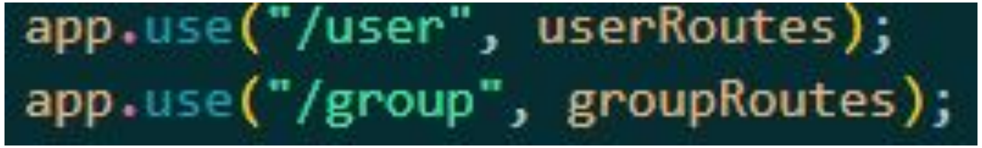
```jsx
function AllGroups() {
  const { classes } = useStyles();
  const { groups } = useSelector((state) => state.groups);

  return (
    <Navbar.Section className={classes.section}>
      <Group className={classes.groupsHeader} position="apart">
        <Text size="s" weight={500} color="dimmed">
          All your groups
        </Text>
      </Group>
      <ScrollArea className={classes.groups} offsetScrollbars>
        {groups.map((group, index) => (
          <GroupLink key={index} data={group} />
        ))}
      </ScrollArea>
    </Navbar.Section>
  );
}
```

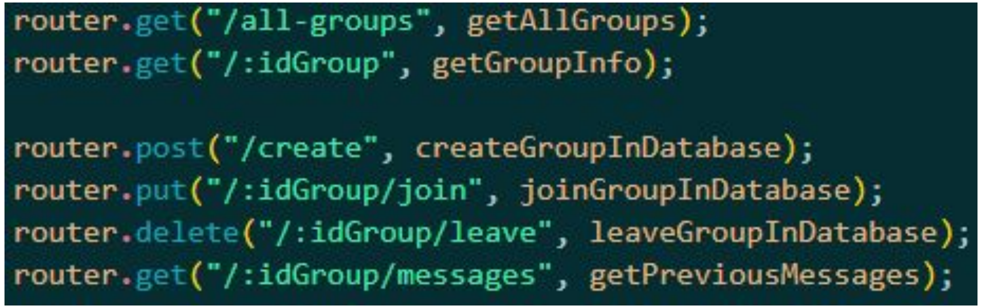


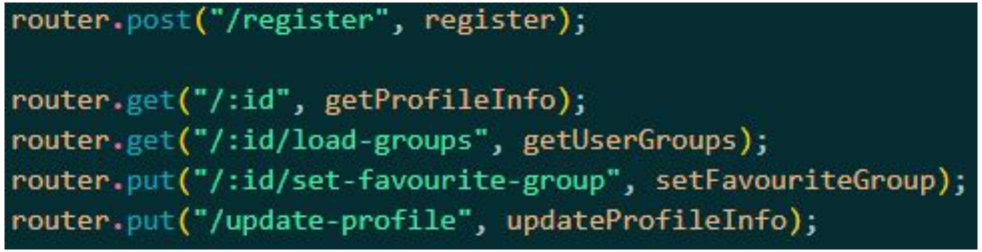
```jsx
function GroupLink({ data }) {
  const { name, imgLink, uid } = data;
  const { classes } = useStyles();
  const navigate = useNavigate();
  const dispatch = useDispatch();

  const handleSelectGroup = () => {
    dispatch(setActiveGroup(data));
    dispatch(setActiveChatGroupAction(uid));
  };
  function handleClick() {
    navigate(PATHS.CHAT, { replace: true });
  }
  return (
    <div style={{ position: "relative" }}>
      <a
        href="/"
        onClick={(e) => {
          e.preventDefault();
          handleSelectGroup();
          handleClick();
        }}
        key={name}
        className={classes.groupLink}>
        <Group>
          <Avatar src={imgLink} radius="xl" />
          {name}

          {/* {notifications && (
            <Badge
```

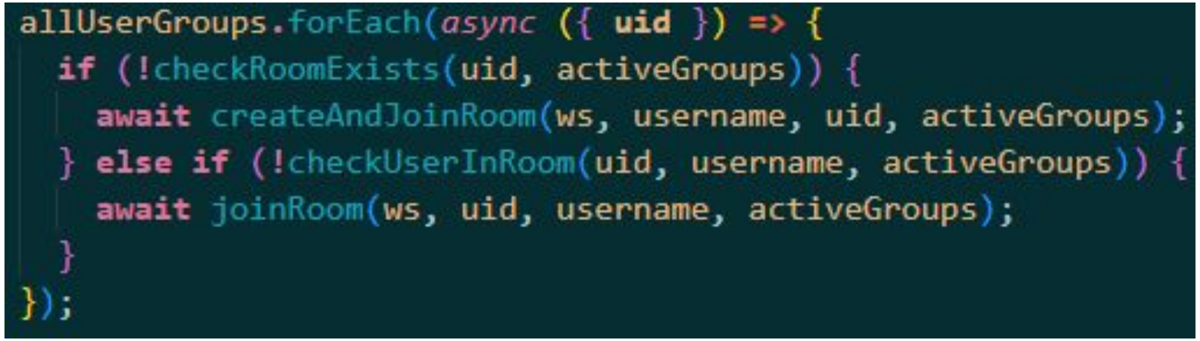
Backend


```
> database
> server
> websocket
  index.js
```

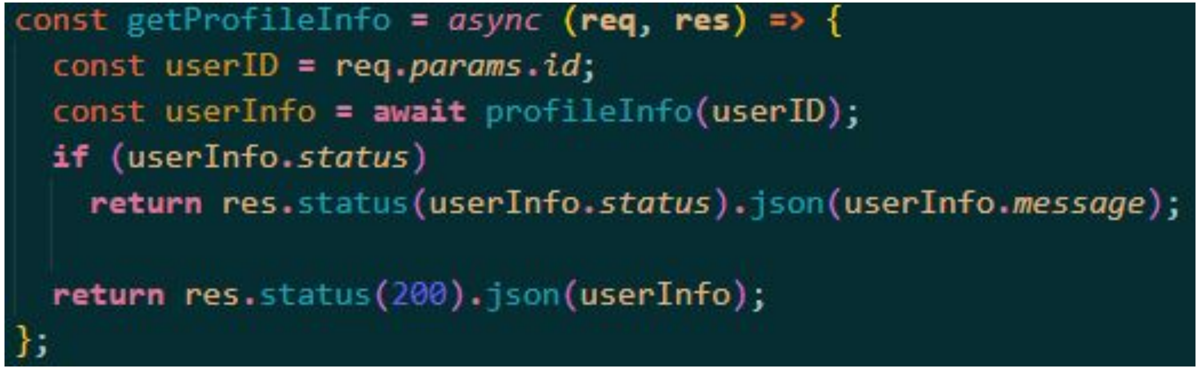

```
app.use("/user", userRoutes);
app.use("/group", groupRoutes);
```


```
router.get("/all-groups", getAllGroups);
router.get("/:idGroup", getGroupInfo);

router.post("/create", createGroupInDatabase);
router.put("/:idGroup/join", joinGroupInDatabase);
router.delete("/:idGroup/leave", leaveGroupInDatabase);
router.get("/:idGroup/messages", getPreviousMessages);
```


```
router.post("/register", register);

router.get("/:id", getProfileInfo);
router.get("/:id/load-groups", getUserGroups);
router.put("/:id/set-favourite-group", setFavouriteGroup);
router.put("/update-profile", updateProfileInfo);
```



```javascript
allUserGroups.forEach(async ({ uid }) => {
  if (!checkRoomExists(uid, activeGroups)) {
    await createAndJoinRoom(ws, username, uid, activeGroups);
  } else if (!checkUserInRoom(uid, username, activeGroups)) {
    await joinRoom(ws, uid, username, activeGroups);
  }
});
```



```javascript
const getProfileInfo = async (req, res) => {
  const userID = req.params.id;
  const userInfo = await profileInfo(userID);
  if (userInfo.status)
    return res.status(userInfo.status).json(userInfo.message);

  return res.status(200).json(userInfo);
};
```