

Міністерство освіти і науки України
Львівський національний університет імені Івана Франка
Факультет електроніки та комп'ютерних технологій

Звіт
про виконання проектної роботи
«Соціальна Мережа (Social)»

Виконав:
студент групи ФЕП-13
Андріан Карсанашвілі
Прийняла:
Ковтко Р.Т.

Львів 2021

Технології: React + Redux, TypeScript, SCSS, React libs.

API: <https://social-network.samuraijs.com/>

Host: <https://andrian-kars.github.io/social/>

GitHub: <https://github.com/andrian-kars>

Виконання роботи

Проект розбитий на 3 рівня:

DAL: Data Access Layer

BLL: Business Logic Layer

UI: User Interface

DAL

Для отримання даних з сервера я використав бібліотеку **axios**.

Заготовка для всіх запитів:

```
export const instance = axios.create({
  baseURL: 'https://social-network.samuraijs.com/api/1.0/',
  withCredentials: true,
  headers: {
    'API-KEY': apiKey,
  },
})
```

Приклад самих запитів:

```
export const dialogsAPI = {
  getDialogs() {
    return instance.get<APIResponseType<Array<DialogType>>>>('dialogs')
      .then(res => res.data) },
  startDialog(userId: number) {
    return instance.put<APIResponseType>(`dialogs/${userId}`)
      .then(res => res.data) },
  getMessages(userId: number) {
    return instance.get<APIResponseType<Array<MessageType>>>>(`dialogs/${userId}/messages`)
      .then(res => res.data) },
  sendMessage(userId: number, body: string) {
    return instance.post<APIResponseType>(`dialogs/${userId}/messages`, {body: body})
      .then(res => res.data) },
}
```

BLL

Для логічної частини я використовую **Redux**.

Store:

```
const rootReducer = combineReducers({
  profilePage: profileReducer,
  dialogsPage: dialogsReducer,
  usersPage: usersReducer,
  auth: authReducer,
  form: formReducer,
  app: appReducer,
  news: newsReducer
})
const store = createStore(rootReducer, applyMiddleware(thunk))
```

Приклад Reducer'a:

```
const initialState = {
  dialogs: [] as Array<DialogType>,
  messages: [] as Array<MessageType>,
  isFetching: false,
  isSubFetching: false
}

const dialogsReducer = (state = initialState, action: ActionsType): InitialStateType => {
  switch (action.type) {
    // Helpers
    case 'S/DIALOGS/SET_IS_FETCHING': {
      return {
        ...state,
        isFetching: action.isFetching,
      }
    }
    case 'S/DIALOGS/SET_SUB_IS_FETCHING': {
      return {
        ...state,
        isSubFetching: action.isSubFetching,
      }
    }
    // Main
    case 'S/DIALOGS/SAVE_DIALOGS': {
      return {
        ...state,
        dialogs: action.dialogs,
      }
    }
  }
}
```

Приклад actions:

```
export const actions = {
  // Helpers
  setIsFetching: (isFetching: boolean) => (
    { type: 'S/DIALOGS/SET_IS_FETCHING', isFetching } as const),
  setIsSubFetching: (isSubFetching: boolean) => (
    { type: 'S/DIALOGS/SET_SUB_IS_FETCHING', isSubFetching } as const),
  // Main
  setDialog: (userId: number, userName: string) => (
    { type: 'S/DIALOGS/START_DIALOG', payload: { id: userId, userName: userName } } as const),
  saveDialogs: (dialogs: Array<DialogType>) => (
    { type: 'S/DIALOGS/SAVE_DIALOGS', dialogs } as const),
  saveMessages: (messages: Array<MessageType>) => (
    { type: 'S/DIALOGS/SAVE_MESSAGES', messages } as const),
  sendMessage: (message: MessageType) => (
    { type: 'S/DIALOGS/SEND_MESSAGE', message } as const),
}
```

Приклад thunk'ки:

```
export const saveProfile = (profile: ProfileType): ThunkType => async (dispatch, getState) => {
  dispatch(actions.setIsFetching(true))
  const userId = getState().auth.userId
  const profileData = await profileAPI.saveProfile(profile)
  if (profileData.resultCode === ResultCodesEnum.Success) {
    if (userId !== null) {
      dispatch(getProfile(userId))
    } else {
      throw new Error('User ID can\'t be null')
    }
  } else {
    dispatch(stopSubmit('editProfile', { _error: profileData.messages[0] }))
    return Promise.reject(profileData.messages[0])
  }
  dispatch(actions.setIsFetching(false))
}
```

Приклад selector'ів:

```
export const getPageSize = (state: AppStateType) => {
  return state.usersPage.pageSize
}

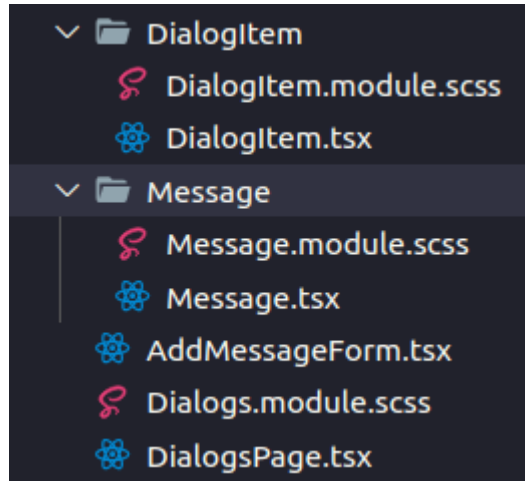
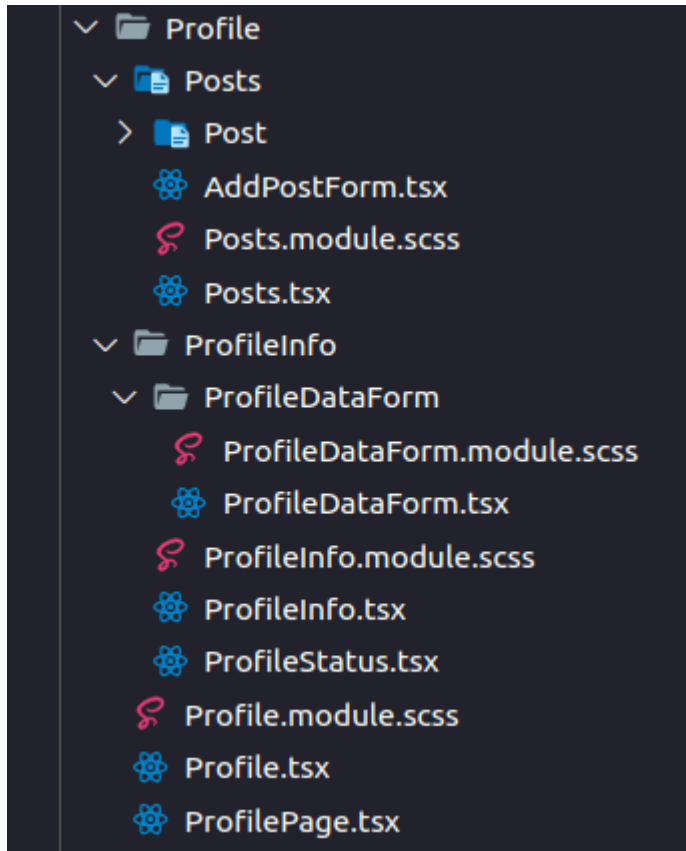
export const getTotalUsersCount = (state: AppStateType) => {
  return state.usersPage.totalUsersCount
}

export const getCurrentPage = (state: AppStateType) => {
  return state.usersPage.currentPage
}
```


UI

Весь UI розбитий на компоненти.

Приклад таких компонентів:



Приклад отримання даних із store:

```
const totalUsersCount = useSelector(getTotalUsersCount)
const currentPage = useSelector(getCurrentPage)
const pageSize = useSelector(getPageSize)
const users = useSelector(getUsers)
const followingInProgress = useSelector(getFollowingInProgress)
const filter = useSelector(getUsersFilter)
const isFetching = useSelector(getIsFetching)
const isSubFetching = useSelector(getIsSubFetching)

// to check and change message button
const dialogs = useSelector(getDialogs)
```

Приклад функцій які міняють store:

```
const onPageChange = (pageNumber: number) => { dispatch(requestUsers(pageNumber, pageSize, filter)) }
const onFilterChange = (filter: FilterType) => { dispatch(requestUsers(1, pageSize, filter)) }
const onFollow = (userId: number) => { dispatch(follow(userId)) }
const onUnfollow = (userId: number) => { dispatch(unfollow(userId)) }
const onStartDialog = (userId: number, name: string) => { dispatch(startDialog(userId, name)) }
const authorizedUserId = useSelector((state: AppStateType) => state.auth.userId)
```

Приклад *jsx*:

```
return (
  <header id="top" className={s.header}>
    <div className={s.wrapperLeftHead}>
      <NavLink to={'/'} className={s.wrapperSocial}>
        <img className={s.logo} src={logo} alt="logo" />
        <h1 className={s.heading}>Social</h1>
      </NavLink>
      <div onClick={toggleMenu} className={s.burgerWrapper}>
        <Burger show={!menu} />
      </div>
    </div>
    <div className={s.loginBlock}>
      <div onClick={toggleShow} className={s.dropDown}>
        <div className={s.login}>
          <span>{loginName}</span>
          <svg viewBox="0 0 20 20">
            <path d="M13.962,8.885l-3.736,3.739c-0.086,0.086-0.201,0.1..."/>
          </svg>
        </div>
        <div className={s.divToShow}>
          <span className={s.logout} onClick={onLogout}>Logout</span>
        </div>
      </div>
    </div>
  </header>
)
```

Приклад утиліти:

```
export const required: FieldValidatorType = value => {
  if (value) return null
  return 'Cannot be empty'
}

export const maxLengthCreator = (maxLength: number): FieldValidatorType => value => {
  if (value && value.length > maxLength) return `Max length is ${maxLength} symbols`
  return null
}
```

Приклад типізації:

```
export type PostType = {
  id: number
  likesCount: number
  message: string
}

export type ProfileType = {
  userId: number
  lookingForAJob: boolean
  lookingForAJobDescription: string
  fullName: string
  contacts: ContactsType,
  photos: PhotosType
  AboutMe?: string
}
```

Функціонал

Login:

Sign in

Enter your email address and password or use free account for test

Email address

Password

☐

Remember me

Sign in

You can login with a test account [here](#)

Присутня валідація, також можна зайти через тестовий аккаунт. Якщо буде багато запитів на сервер, появиться каптча.

Navigation:



News



Profile



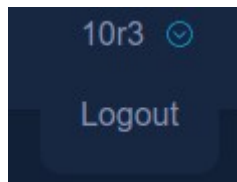
Messages



Users

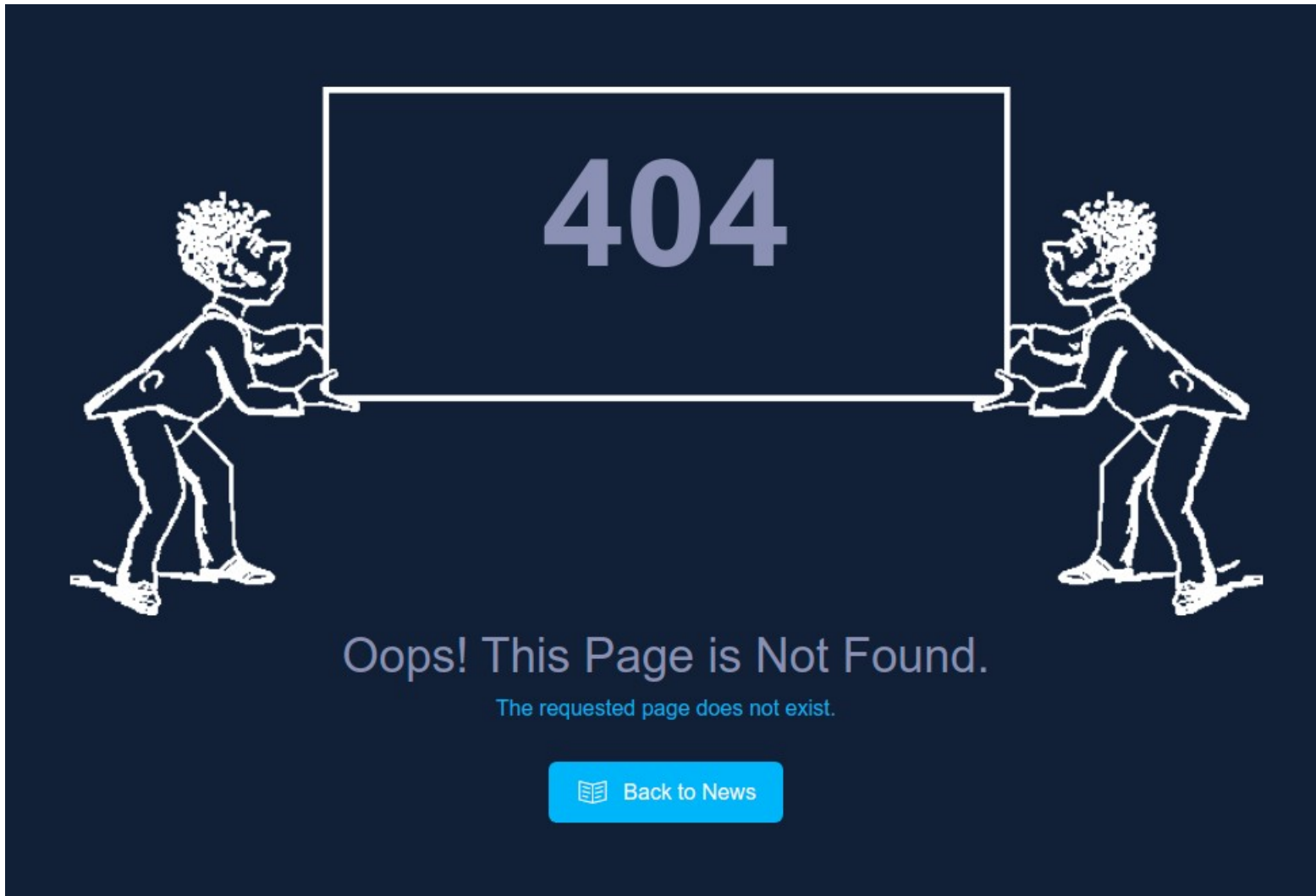
Навігація серед різних сторінок.

Header:



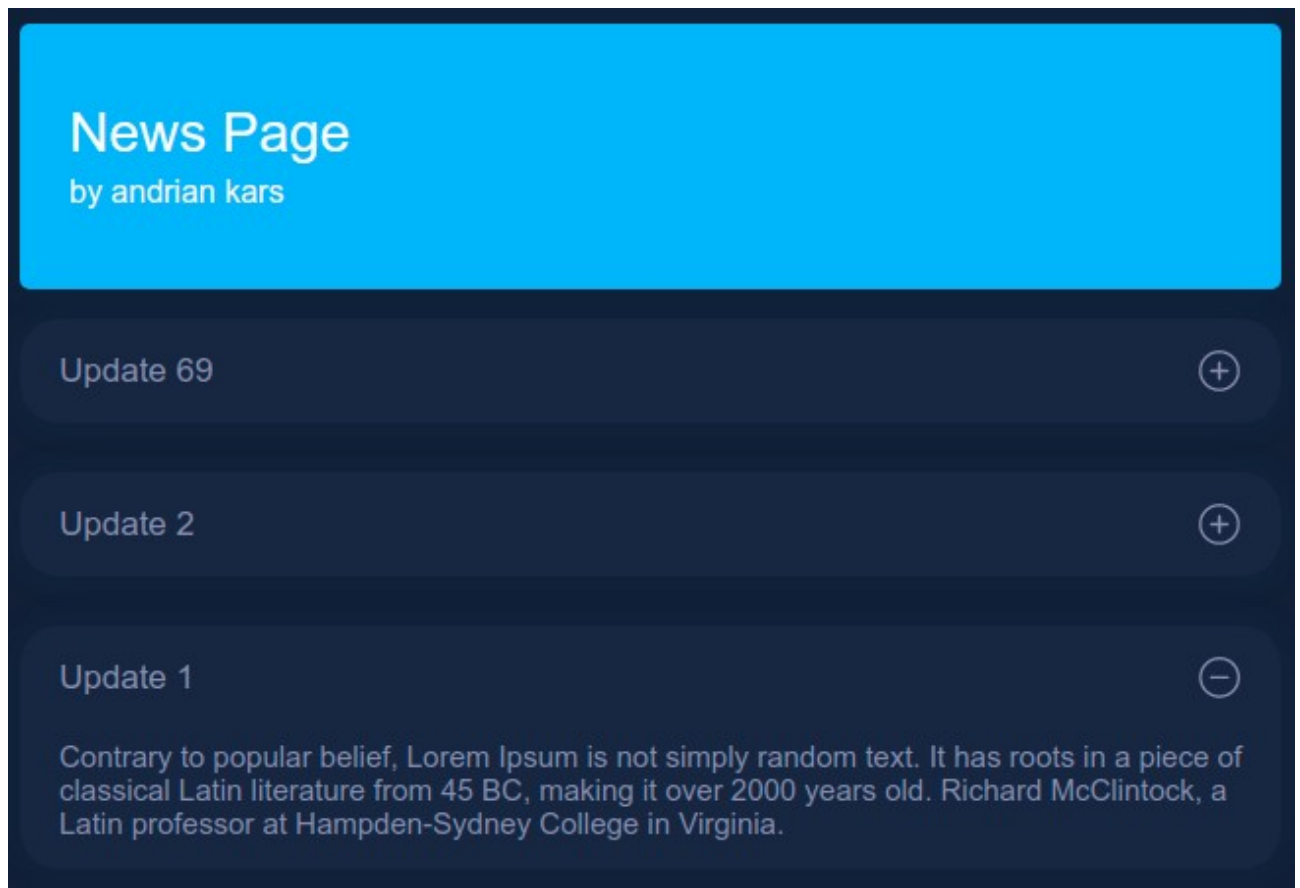
Силка, яка веде на news сторінку. Випадаючий список, він дає можливість вилогінитися. Також туди буде добавлено зміну мови та теми.

404 Page:



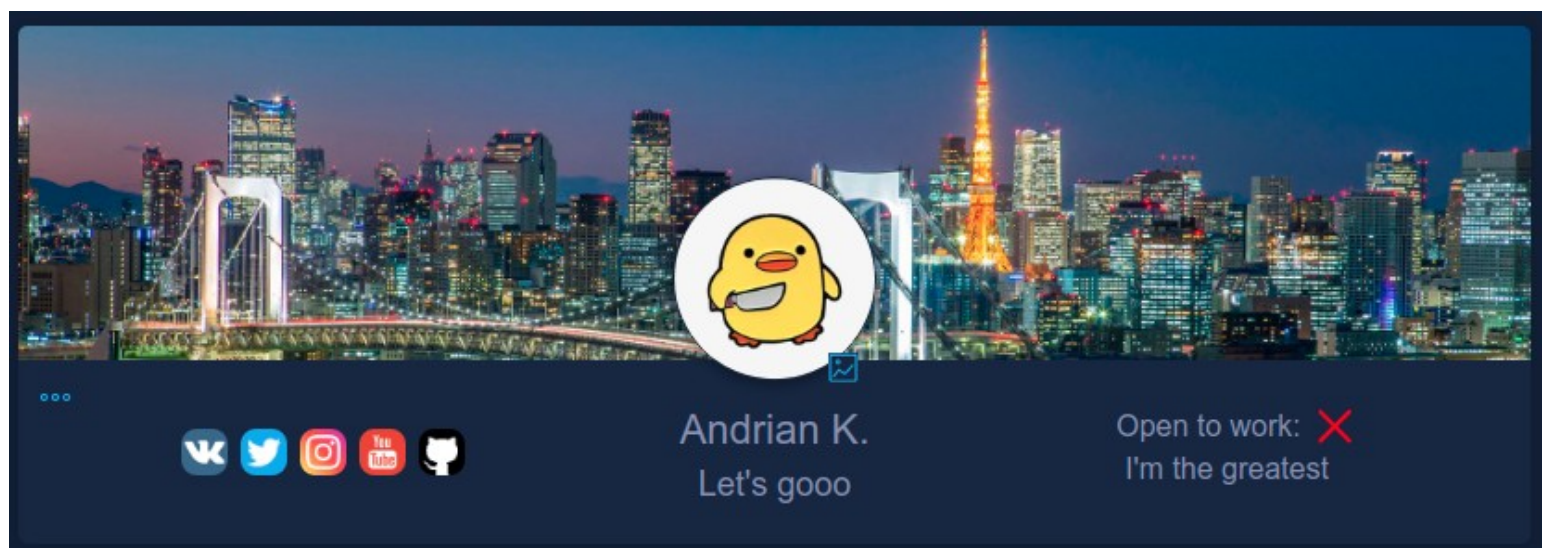
Якщо введений не правильний url адрес, тоді появиться така сторінка.

News Page:

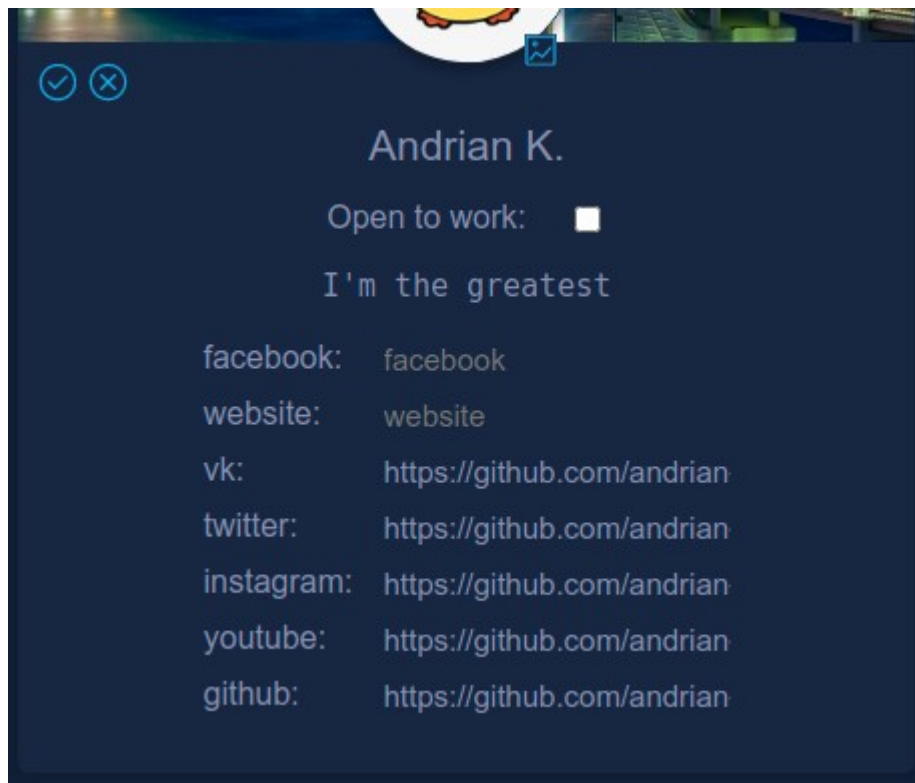


Тут будуть добавлятися новини по оновленням проекту. Новини з анімацією та можна побачити тільки одну за раз.

Profile Page:



Персональна сторінка, тут можна міняти аватарку та статус. Якщо клікнути на три крапки, відкриється вікно редагування решти інформації.



Andrian K.

Open to work: ☐

I'm the greatest

facebook: facebook

website: website

vk: <https://github.com/andrian>

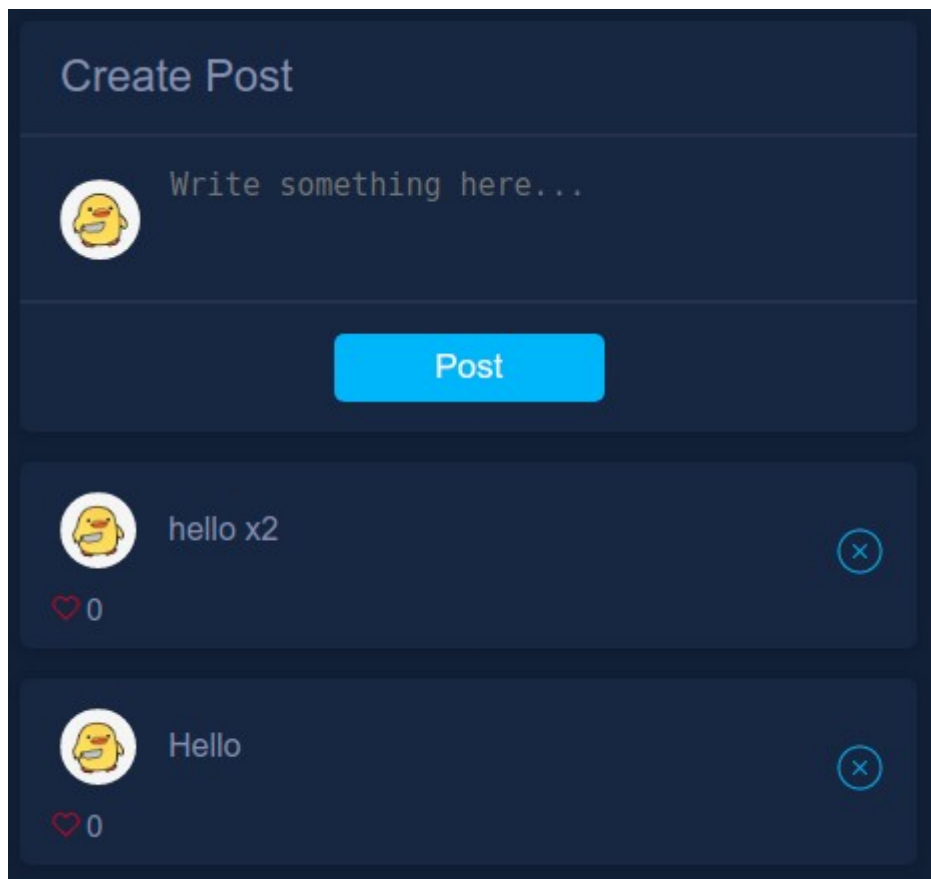
twitter: <https://github.com/andrian>

instagram: <https://github.com/andrian>

youtube: <https://github.com/andrian>

github: <https://github.com/andrian>

Тут можна змінити ім'я, готовність до роботи, підпис до роботи та посилання.
Є можливість зберегти зміти та відмінити їх.



Create Post

Write something here...

Post

hello x2

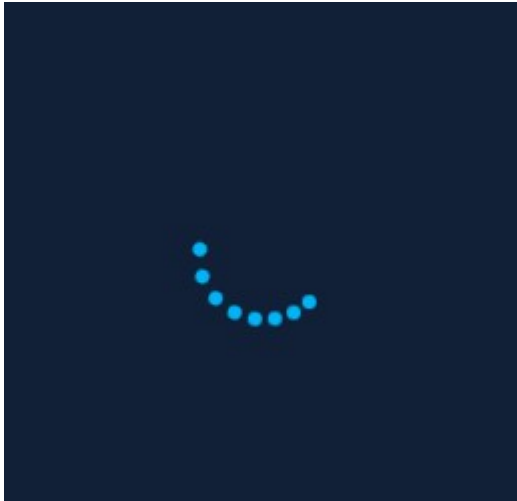
0

Hello

0

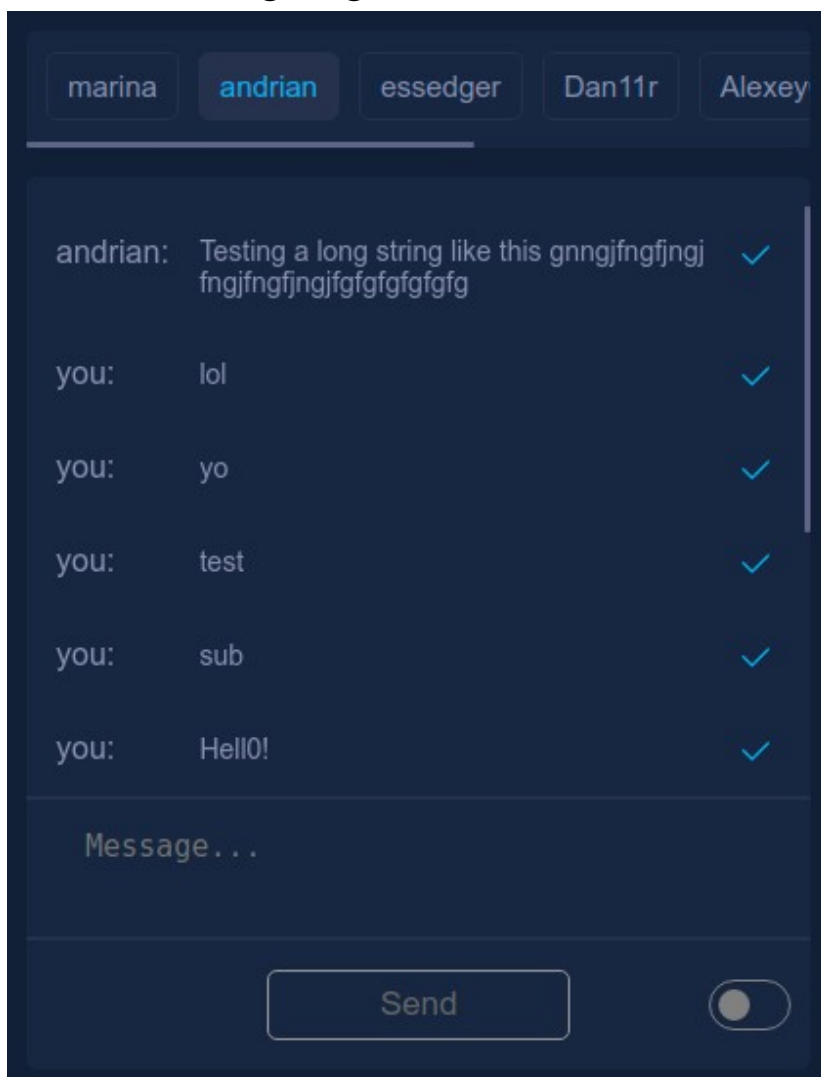
Пости. Їх можна добавляти та видаляти. Пости не підтримуються API, тому я їх адаптував через LocalStorage. При добавленні нового поста присутня валідація.
Якщо постів немає, покажиться placeholder.

Loader:

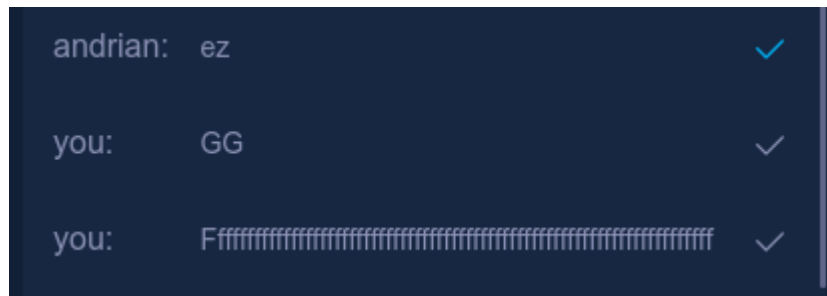


Якщо на сайті щось змінилось, або загрузається, то буде показана така загрузка.

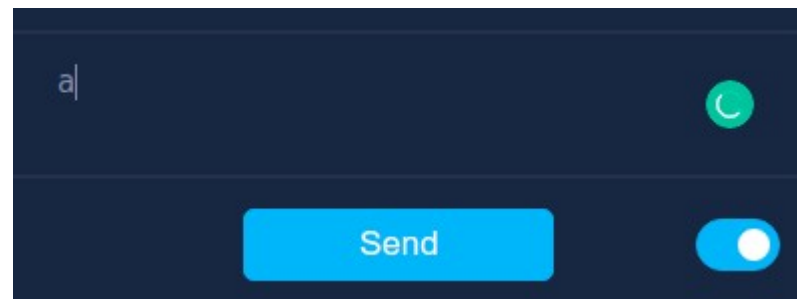
Dialogs Page:



Сторінка діалогів. Якщо немає співрозмовника, або він не вибраний, то покажиться placeholder

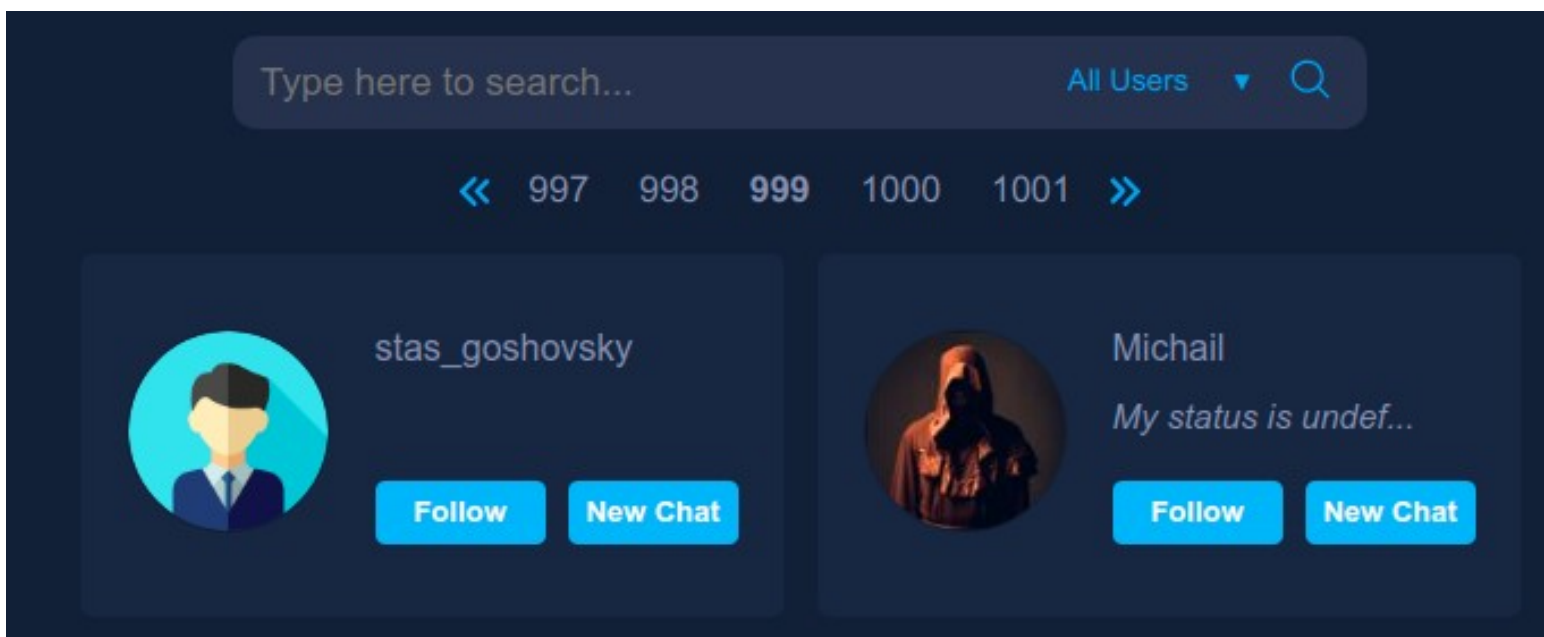


Якщо оба співрозмовника прочитали повідомлення, то галочка буде синьою.

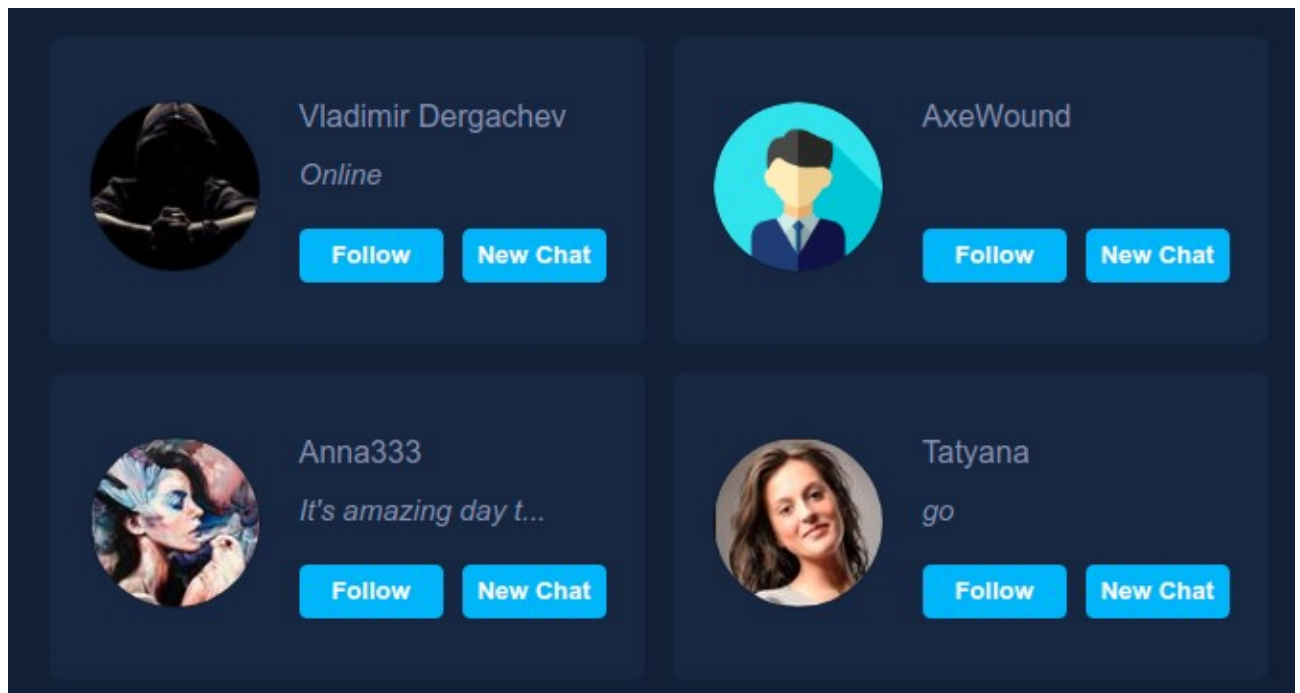


Присутня валідація, та Live Reload типу web socket'а. Якщо валідація не пройдена, кнопка буде сірою.

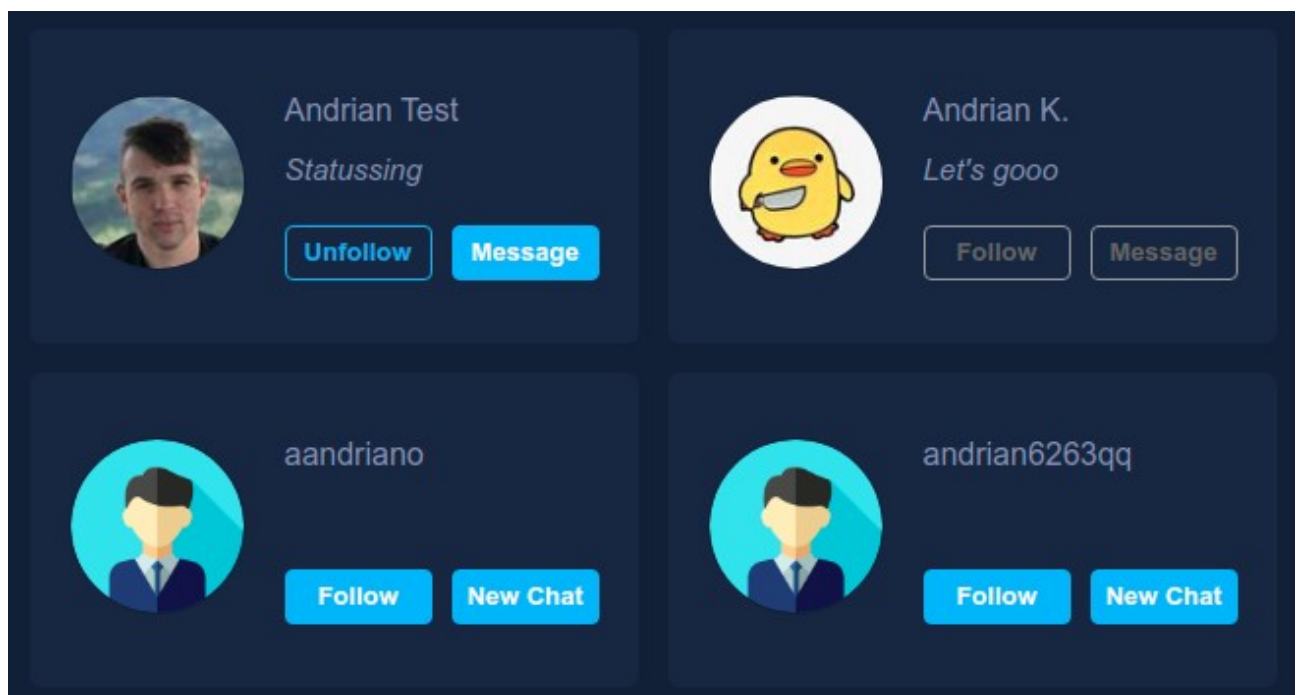
Users Page:



Сторінка, де видно всіх користувачів. Можна зробити пошук по імені, вибрати людей на яких ви підписані, або не підписані. Також присутня пагінація. По п'ять сторінок за раз, також є кнопки для преміщення на останню сторінку і на першу.

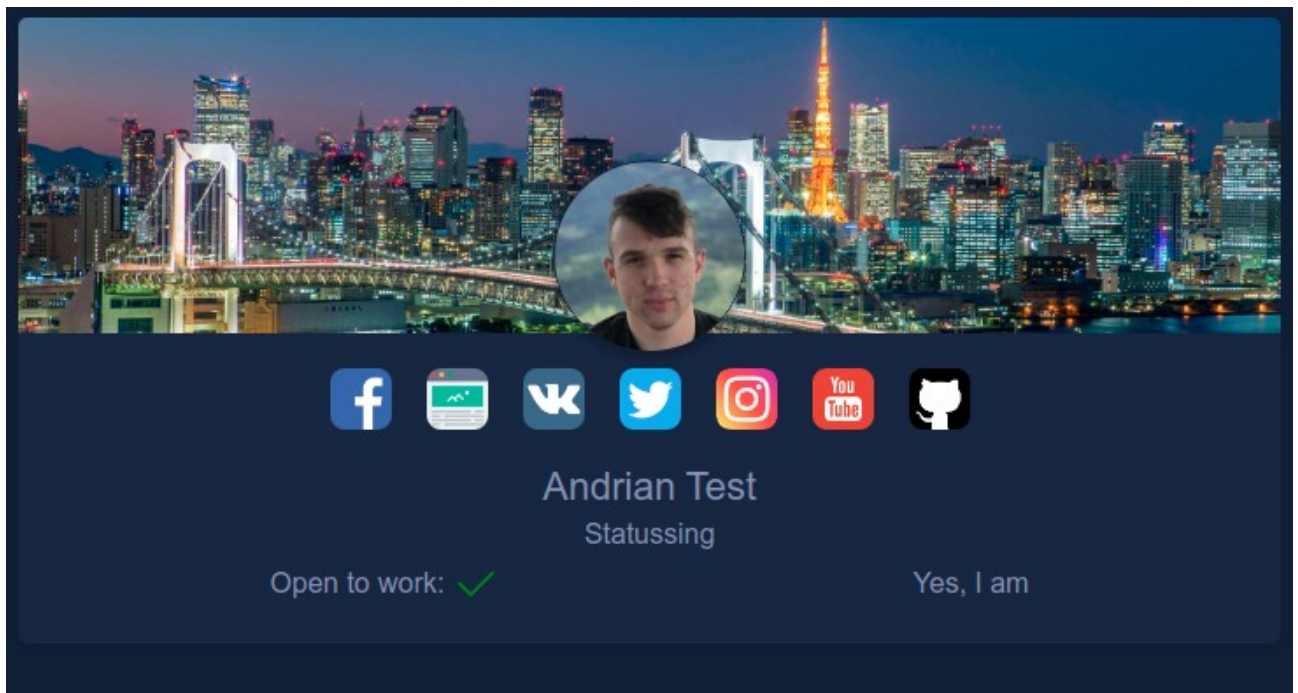


На одній сторінці показується 12 юзерів. Якщо ім'я, або статус дуже великі, то будуть скорочені, але при наведенні можна побачити їх повністю.



Присутні дві кнопки. Слідкувати та почати спілкування. Якщо ви знайшли себе, то обі кнопки не будуть активними. Якщо ви вже почали з людиною чат, то текст кнопки поміняється на 'Message'. Якщо ви почали слідкувати, то кнопка також зміниться. При кліку на другу кнопку перекине на сторінку діалогів до вибраного користувача. Якщо у користувача немає фото, то покажеться placeholder. При кліку на аватарку перенаправить на сторінку користувача.

Profile Page (не основного користувача):



На даній сторінці немає постів, так як вони не підтримуються API та нічого не можна редагувати.

- Все на проєкті ідеально оптимізовано. Більше половини часу я потратив саме на оптимізацію.
- Сайт ідеально адаптивний, на будь-якому пристрої все буде працювати добре.
- В майбутньому буде додано багато нових фіч.
- Під час всієї розробки була використана система контролю версій git. Тому зараз більше 200 комітів до яких можна повернутися.
- Доданий README файл, де вказано основну інформацію та як запустити проєкт іншому користувачу.
- Це мій перший проєкт на React'ті. Я спочатку розробив сайт на старих можливостях React'ту, а потім переніс все на саме свіже.
- Проєкт повністю типізований.
- Для виконання цієї роботи я потратив більше 800 годин та дуже добре засвоїв React.

Висновок: Реалізована соціальна мережа, добре оптимізована, типізована та на нових технологіях. Також присутня величезна кількість маленьких фіч.