Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Лабораторна робота №4

Дисципліна: «Архітектура програмного забезпечення»

Виконав:
студент групи ПЗПІ-21-10
Готвянський Кирило Павлович

Перевірив:
ст. викл. каф. ПІ
Сокорчук Ігор Петрович

Харків 2024

ВСТУП

Метою роботи є розробка клієнтської частини проєкту за темою: «Програмна система для моніторингу та аналізу відвідувачів басейну».

Хід даної лабораторної роботи має такий вигляд:

1. Опис прийнятих інженерних рішень;
2. Загальна архітектура системи;
3. Взаємодія з іншими частинами системи.

1 МОДЕЛЮВАННЯ ПРОГРАМНОЇ СИСТЕМИ

Користувачами клієнтської частини розробленої системи є 2 типи акторів: адміністратор та член басейну.

Основними потребами клієнта є: можливість подавати заяву у басейн, дивитись свій профіль, бачити свої здані виміри, отримувати рекомендації згідно вимірів

Основними потребами адміністратора є: адміністрування басейну, адміністрування користувачів, експортування/імпортування даних.

Взаємодію кожної ролі користувачів з клієнтською частиною системи представлено на діаграмі прецедентів (див. додат. А.1).

Таким чином, за допомогою діаграми прецедентів було визначено функціональні потреби та взаємодію різних типів користувачів із системою.

2 ТЕХНОЛОГІЇ ТА АРХІТЕКТУРНІ РІШЕННЯ

Для написання клієнтської частини системи було обрано мову програмування TypeScript з використанням технології React, що забезпечує високу швидкодію розробки і зручність у використанні за рахунок статичної типізації та компонентного підходу.

Для реалізації елементів інтерфейсу було використано бібліотеку Material UI, яка надає готові та стилізовані компоненти інтерфейсу для швидкої і зручної розробки візуально привабливого веб-додатку.

Для реалізації локалізації інтерфейсу було використано бібліотеку i18next, що дозволяє легко і ефективно впроваджувати багатомовний інтерфейс та забезпечує зручний механізм перекладу текстових рядків.

Для реалізації навігації по сайту використано бібліотеку react-router-dom, яка надає зручні та потужні інструменти для створення реактивної навігації та управління маршрутами веб-додатку.

Для виконання асинхронних запитів на сервер використано бібліотеку axios, яка забезпечує зручний та потужний інтерфейс для взаємодії з HTTP-запитами та дозволяє легко обробляти відповіді від сервера.

Для розгортання серверу використано інструмент побудови Vite, який надає швидкий та ефективний спосіб розгортання веб-додатків, дозволяючи легко налаштовувати середовище розробки та підтримує автоматичне оновлення в реальному часі.

Для візуалізації використаних пакетів було розроблено діаграму пакетів (див. рис. 1).

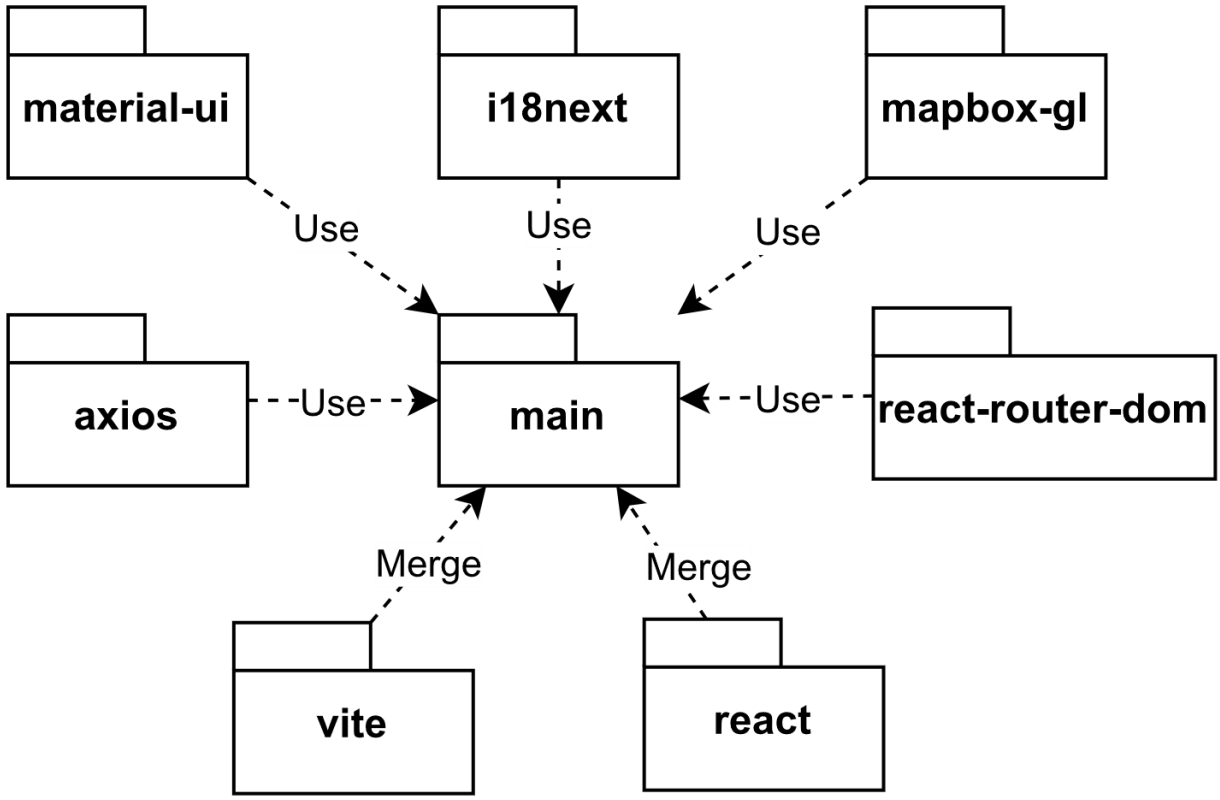

Рисунок 1 – Діаграма пакетів для клієнтської частини системи «Lieb ist Pool»

Розроблено клієнтську частину за допомогою Feature-Sliced архітектури. Вона визначає побудову проєкта за допомогою шарів.

Шари включають зрізи, які в свою чергу включають сегменти. Шари – це каталоги верхнього рівня застосунку. У проєкті визначено 7 шарів: app, pages, components, features, interfaces, context та hooks. Шар app визначає основну конфігурацію застосунку. Шар pages містить сторінки сайту. Шар components містить різні компоненти інтерфейсу, які можна перевикористовувати. Шар features відповідає за функціонал застосунку. Він визначає дії, доступні користувачу в системі. Шар interfaces визначає інтерфейси моделей, які використовуються для обміну даними з сервером та їхнього представлення на сторінках. Шар context визначає об’єкти контексту, які дозволяють передачу даних іншим компонентам, які відповідають контексту. Шар hooks відповідає за об’єкти хуків, які дозволяють компонентам отримувати доступ до станів системи.

Для більш детальної візуалізації архітектури системи було розроблено діаграму компонентів (див. додат. Б.1).

Взаємодія користувача з клієнтською частиною починається з введення даних профілю (при реєстрації або входу). Після авторизації, згідно ролі користувача, взаємодія розділяється на клієнтську та адміністраторську.

Клієнт має можливість записатись у басейн. Для цього він у своєму профілі обирає доступний басейн зі списку та очікує схвалення або відхилення заяви від адміністратору басейну. Також клієнт може отримувати рекомендації за вимірами, які він здав у басейні.

Адміністратор має можливість керування басейнами, набирати членів до басейну та експортувати дані.

Для більш детальної візуалізації взаємодії користувачів з клієнтської частиною системи було розроблено діаграму взаємодії (див. додат. В.1).

Таким чином, за допомогою діаграм пакетів, компонентів та взаємодії було визначено основні технології та архітектуру системи.

Приклади коду клієнтської частини системи представлено у додатку Г.

ВИСНОВКИ

У ході виконання лабораторної роботи були отримані навички з проєктування клієнтської частини програмної системи. Результатом роботи є розроблена програмна реалізація клієнтської частини програмної системи з використанням React. При розробці продукту дотримувались вимоги до чистого коду мови програмування Typescript.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Посилання на відео захисту. URL: https://youtu.be/guy0RKh-uSw

2. Посилання на папку з джерельним кодом проєкта. URL: https://github.com/NureHotvianskyiKyrylo/apz-pzpi-21-10-hotvianskyi-kyrylo/tree/main/Task4.

ДОДАТОК А

Діаграма прецедентів

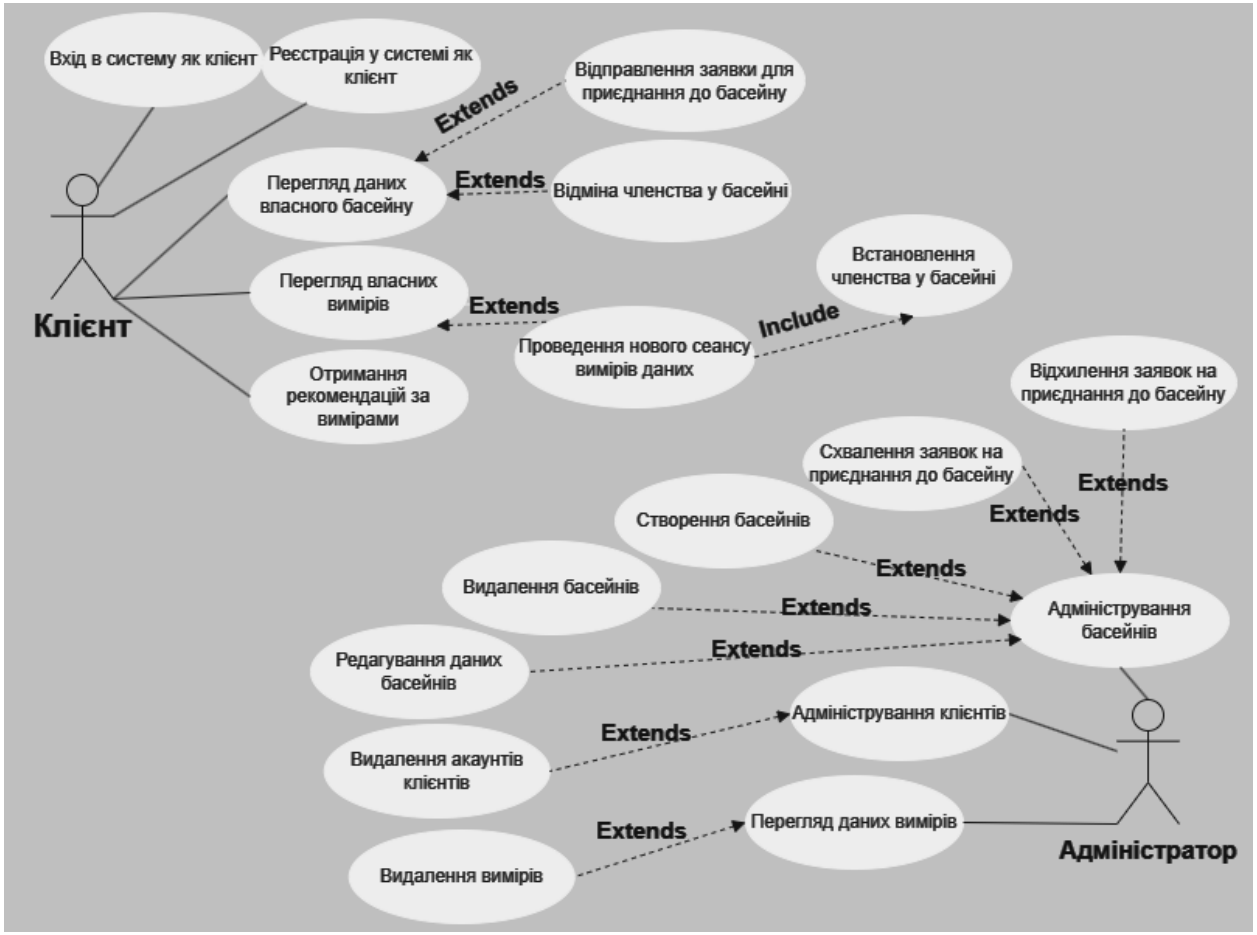

Рисунок А.1 – Діаграма прецедентів клієнтської частини системи «Lieb ist Pool»

ДОДАТОК Б

Діаграма компонентів

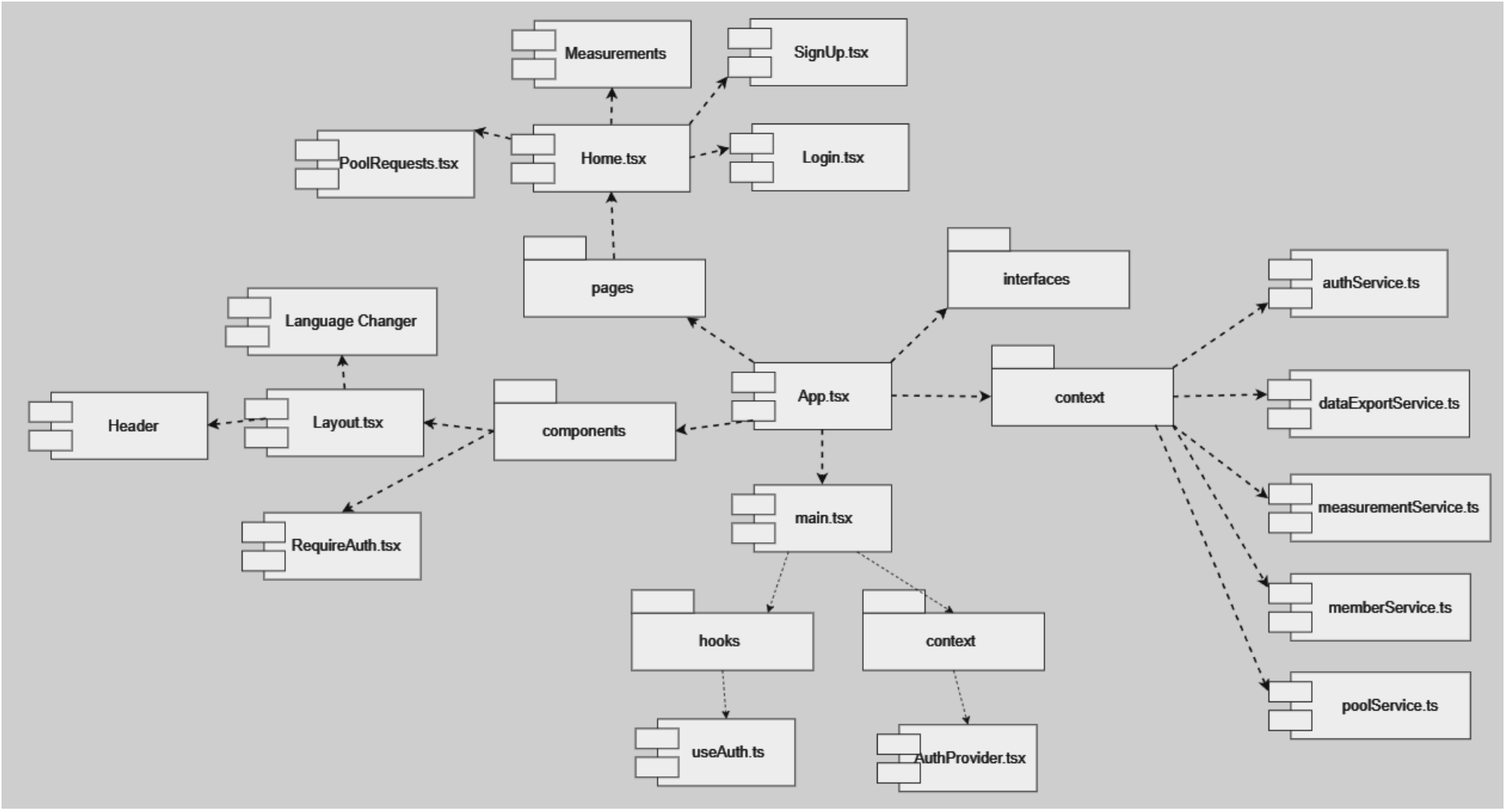

Рисунок Б.1 – Діаграма компонентів для клієнтської частини системи «Lieb ist Pool»

ДОДАТОК В

Діаграма взаємодії

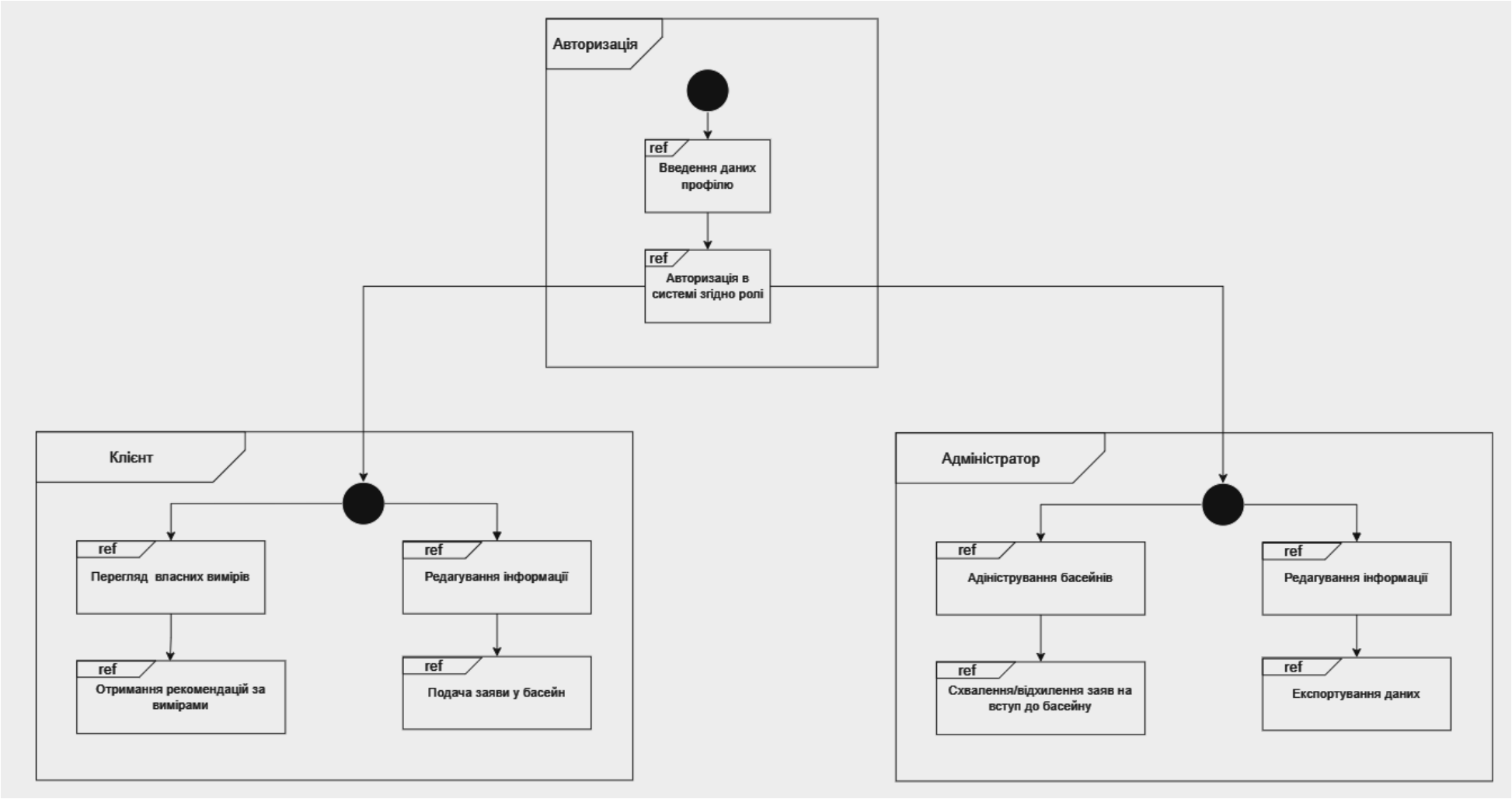

Рисунок В.1 – Діаграма взаємодії для клієнтської частини системи «Lieb ist Pool»

ДОДАТОК Г

Програмний код клієнтської частини

Г.1 Код сервісу для відправлення запитів користувача

```
1 import axios from "axios";
2 import apiClient from "../config/apiClient";
3     import     {MemberDto,     UpdateMemberCommand}     from
"../interfaces/member.ts";
4
5 const updateMember = async (
6     token : string,
7     updateMemberCommand : UpdateMemberCommand
8 ) : Promise<void> => {
9     try {
10         const headers = {
11             'Authorization': 'Bearer ' + token
12         };
13         await apiClient.put(
14             'members/updateInfo',
15             updateMemberCommand,
16             { headers }
17         );
18         return;
19     } catch (error) {
20         if (axios.isAxiosError(error)) {
21             throw new Error(error.response?.data.message);
22         } else {
23             throw new Error("Unknown error occurred.");
24         }
25     }
26 }
27
28 const getMemberByUser = async (
```

```
    token : string
) : Promise<MemberDto> => {
    try {
        const headers = {
            'Authorization': 'Bearer ' + token
        };
        const response = await apiClient.get<MemberDto>(
            'members/getByUser',
            { headers }
        );
        return response.data;
    } catch (error) {
        if (axios.isAxiosError(error)) {
            throw new Error(error.response?.data.message);
        } else {
            throw new Error("Unknown error occurred.");
        }
    }
}

const enrollMemberToPool = async (
    token : string,
    memberId : number,
    poolId : number
) : Promise<void> => {
    try {
        const headers = {
            'Authorization': 'Bearer ' + token
        };
        await apiClient.put(
            'members/enrollToPool',
            { memberId, poolId },
            { headers }
        );
```

```
        return;
    } catch (error) {
        if (axios.isAxiosError(error)) {
            throw new Error(error.response?.data.message);
        } else {
            throw new Error("Unknown error occurred.");
        }
    }
}

export const membersService = {
    updateMember,
    getMemberByUser,
    enrollMemberToPool
};
```

Г.2 Код сторінки вимірів користувача

```
import {
    Box,
    Container,
    IconButton,
    Paper,
    Table,
    TableBody,
    TableCell,
    TableHead,
     TableRow,
     Typography
} from "@mui/material";
import {useTranslation} from "react-i18next";
import useAuth from "../hooks/useAuth.ts";
```

```
15 import InfoIcon from '@mui/icons-material/Info';
16 import {useEffect, useState} from "react";
17 import {measurementsService} from "../features/measurementsService.ts";
18 import {MeasurementDto} from "../interfaces/measurement.ts";
19 import {RecommendationDto} from "../interfaces/recommendation.ts";
20
21 export default function Measurements() {
22     const { t } = useTranslation();
23     const { auth } = useAuth();
24     const [measurements, setMeasurements] = useState<MeasurementDto[]>();
25     const [recommendations, setRecommendations] = useState<RecommendationDto[]>()
26
27     useEffect(() => {
28         const fetchMeasurements = async () => {
29             try {
30                 const response = await measurementsService.getAllMeasurementsByMember(auth.bearer!);
31                 setMeasurements(response);
32             } catch (error) {
33                 console.error('Error fetching pools:', error);
34             }
35         };
36         fetchMeasurements();
37     }, [auth.bearer]);
38
39    const handleGetRecomendation = async (measurementId: number) => {
40         try {
```

```
41              const recommendations : RecommendationDto[] = await
measurementsService.getRecommendationByIdForMember(auth.bearer!,
measurementId);
42              setRecommendations(recommendations)
43          } catch (error) {
44              console.error('Error fetching pools:', error);
45          }
46      }
47
48      return (
49          <Container>
50             <Typography variant="h5" gutterBottom align="center"
mt={3} mb={2}>
51                  {t('measurements')}
52              </Typography>
53
54                 <Paper elevation={3} style={{ padding: '20px',
paddingBottom: '20px' }}>
55                  {measurements?.length === 0 && (
56                      <>
57                           <Typography variant="h5" gutterBottom
align="center" mt={3} mb={2}>
58                              {t('noMeasurements')}
59                          </Typography>
60                           <Typography variant="h6" gutterBottom
align="center" mt={3} mb={2}>
61                              {t('makeOneMeasurement')}
62                          </Typography>
63                      </>)}
64                  <Table sx={{ mb: 2 }}>
65                      <TableHead>
66                          <TableRow>
67                              <TableCell>{t('date')}</TableCell>
68                            <TableCell>{t('height')}</TableCell>
```

```
69                              <TableCell>{t('weight')}</TableCell>
70
<TableCell>{t('fatPercentage')}</TableCell>
71
<TableCell>{t('musclePercentage')}</TableCell>
72                             <TableCell>{t('pressure')}</TableCell>
73                                 <TableCell>{t('bmi')}</TableCell>
74
<TableCell>{t('levelOfStress')}</TableCell>
75                                 <TableCell></TableCell>
76                             </TableRow>
77                         </TableHead>
78                         <TableBody>
79                             {measurements?.map((measurement, index)
=> {
80                                                 const date = new
Date(measurement.dateAndTime);
81                                             const dataParseOptions:
Intl.DateTimeFormatOptions = { year: 'numeric', month: 'numeric', day:
'numeric', hour: 'numeric', minute: 'numeric', second: 'numeric' };
82                                     return <TableRow key={index}>
83                                                     <TableCell>{new
Intl.DateTimeFormat('en-US',
dataParseOptions).format(date)}</TableCell>
84
<TableCell>{measurement.height}</TableCell>
85
<TableCell>{measurement.weight}</TableCell>
86
<TableCell>{measurement.fatPercentage}</TableCell>
87
<TableCell>{measurement.musclePercentage}</TableCell>
```

```
88 <TableCell>{`${measurement.upperPressure}/${measurement.lowerPressure}`}</TableCell>
89 <TableCell>{measurement.bodyMassIndex}</TableCell>
90 <TableCell>{measurement.levelOfStress}</TableCell>
91                                         <TableCell>
92                                             <Box display="flex" flexDirection="row">
93                                                 <IconButton
94                                                     color="primary"
95                                                     aria-label={t('addToCart')}
96                                                     onClick={() =>
97                                                 handleGetRecomendation(
98                                                     measurement.id
99                                                 )
100                                                }
101                                                >
102                                                    <InfoIcon/>
103                                                </IconButton>
104                                            </Box>
105                                        </TableCell>
106                                    </TableRow>
107                                }
108                            )}
109                        </TableBody>
110                    </Table>
111
112                            <Typography variant="h5" gutterBottom align="center" mt={3} mb={2}>
113                                                {recommendations && recommendations.map((recommendation) => {
```

```
                                return (
                                    <Typography variant="h6" gutterBottom align="center" mt={3} mb={2}>
                                        {t(recommendation.key)}
                                    </Typography>
                                )
                            })
                        }
                    </Typography>
                </Paper>
            </Container>
        )
}
```

Г.3 Код сторінки логування

```
import { Button, Container, Link, Paper, TextField, Typography } from '@mui/material';
import { useState } from 'react';
import { useNavigate } from 'react-router';
import authService from '../features/authService';
import useAuth from '../hooks/useAuth';
import { AuthResultDto } from '../interfaces/account';
import { useTranslation } from 'react-i18next';

const Login = () => {
  const { setAuth } = useAuth();
  const { t } = useTranslation();

  const navigate = useNavigate();

  const [email, setEmail] = useState<string>('');
```

```
16  const [password, setPassword] = useState<string>('');
17
18  const handleLogin = async () => {
19    try {
20                    const   result:   AuthResultDto   =   await
authService.signIn(email, password);
21      setAuth(result);
22      saveToLocalStorage(result);
23      navigate("/");
24    } catch (error) {
25      console.error('Error');
26    }
27  };
28
29  const saveToLocalStorage = async (authResult : AuthResultDto)
=> {
30    localStorage.setItem('accessToken', authResult.bearer!);
31    localStorage.setItem('userId', authResult.userId!);
32    localStorage.setItem('role', authResult.role!);
33  }
34
35  return (
36     <div className="bg-light pb-5" style={{ minHeight: "100vh"
}}>
37      <Container component="main" maxWidth="xs">
38       <Paper elevation={3} style={{ padding: '30px', marginTop:
'50px' }}>
39          <Typography variant="h5" gutterBottom align="center">
40            {t('signIn')}
41          </Typography>
42          <form>
43            <TextField
44              label={t('email')}
45              variant="outlined"
```

```
46              margin="normal"
47              fullWidth
48              value={email}
49              onChange={(e) => setEmail(e.target.value)}
50            />
51            <TextField
52              label={t('password')}
53              variant="outlined"
54              margin="normal"
55              fullWidth
56              type="password"
57              value={password}
58              onChange={(e) => setPassword(e.target.value)}
59              sx={{mb: 2}}
60            />
61            <Link href="/sign-up">
62              {t('createAnAccount')}
63            </Link>
64            <Button
65              variant="contained"
66              color="primary"
67              fullWidth
68              style={{ marginTop: '20px' }}
69              onClick={handleLogin}>
70              {t('signIn')}
71            </Button>
72          </form>
73        </Paper>
74      </Container>
75    </div>
76  );
77 };
78
79 export default Login;
```