Міністерство науки та освіти України
Харківський національний університет радіоелектроніки

Звіт до лабораторної роботи №3
з дисципліни «Аналіз та рефакторінг коду програмного забезпечення»
На тему: «Програмна система для виявлення задимлення у приміщенні та допомоги при евакуації»

Виконав:
ст. гр. ПЗПІ-19-3
Логвінов О. В.

Перевірив:
ст. викл. каф. ПІ
Сокорчук І.П

Харків 2021

Тема роботи: Розробка клієнтської частини програмної системи.

Мета роботи: Розробити клієнтську/frontend частину використовуючи Angular, а саме: створити розмітку для відображення предметної області, організувати отримання та відправку даних з серверу, передбачити локалізацію та інтернаціоналізацію користувацького інтерфейсу, забезпечити підтримку сертифікатів.

Посилання на файл з кодом на гугл-диску:

https://drive.google.com/file/d/1xVaEr7XdB-wpVL7FEVFldxXGBlE7PQCK/view?usp=sharing

MD5 хеш для pzpi-19-3-lohvinov-oleksandr-lab3 3.zip:
462e64106232ee9bf199f2edd0e9e6e3

Хід роботи:

1) Створення angular-компонентів для відображення предметної області, створення маршрутизації для сайту та обмежень для цих маршрутів
2) Написання http сервісів, що використовують та реалізують бізнес логіку програмної системи
3) Реалізація захисту системи шляхом впровадження сертифікатів та використовування JWT-токену підчас запитів до сервера.
4) Локалізація та інтернаціоналізація інтерфейсу користувача
5) Побудова UML діаграм: UML діаграму прецедентів (Use case diagram), UML діаграму компонентів (Component Diagram), UML діаграму взаємодії (Interaction Overview Diagram), UML діаграму пакетів (Package Diagram);

Для відображення послуг та дій, які система може виконувати, була розроблена Use Case діаграма (див. додаток А). Згідно з цією діаграмою, видно, що акторами можуть бути наступні користувачі: адміністратор, зареєстрований користувач та користувач, що є відповідальною особою у будівлі. Також на

діаграмі існує роль незареєстрованого користувача або гостя. Вони можуть лише виконати реєстрацію у системі.

Що стосується адміністратора, то він може створювати резервні копії бази даних та відновлювати її у разі виникнення проблем. Крім того, адміністратор повинен реєструвати нові пристрої у базі даних, які згодом використовуються у приміщенні та змінювати інформацію про будь який будинок. До того ж, адміністратору доступні усі дії, що і доступні авторизованим користувачам.

У авторизованого користувача є дві ролі – звичайний користувач та відповідальна людина. Змінити роль на відповідальну людину можу лише адміністратор системи або існуюча відповідальна людина. Звичайний користувач може тільки змінити свої персональні дані. Для відповідальної людини доступний значно більший функціонал. Вона може виконувати наступні дії:

- Додавати іншу відповідальну людину для поточної будівлі або забирати права у поточної відповідальної людини
- Змінювати інформацію про будівлю, а саме:
    a) Змінити або додати інформацію про поверх
    b) Змінити або додати інформацію про кімнату на поверсі
    c) Редагувати маршрут евакуації для певного приміщення
    d) Встановлення масштабу для зображення приміщення (внесення даних, у яких ставиться у відповідність координати на зображені приміщення та світові координати)
    e) Перегляд місцезнаходження користувачів, що знаходяться у приміщенні, а також їх контактної інформації як ім'я, прізвище та номер телефону.
    f) Додавання розумного пристрою до приміщення.
    g) Додати інформацію про перевірочний тест для входження у приміщення
- Включити/виключити сигнал тривоги для будівлі

Для зображення того, з яких елементів складається система, було створено діаграму компонентів (див. додаток Б). Зі схеми видно, що основним файлом є index.html. У ньому відображається весь контент сайту, а також зазначаються глобальні стилі, які потрібні для багатьох окремих компонентів. Так як сторінка авторизації та реєстрації мають відмінний дизайн від основного контенту сайту, було розроблено два окремих модуля – app.module та mainContent.module, у кожному з яких визначені маршрути до кожного з компонентів, що належать до певного модуля – appRouter.module та mainContnetRouter.module (див. додаток Д). Для забезпечення захисту деяких маршрутів від користувачів, що не мають певної ролі або не авторизувалися, були створені окремі компоненти – Route Guards, а саме:

- *login.guard* відповідає за перевірку того, чи залишились дані про вхід з попереднього разу. Якщо дані залишилися (JWT-токен) користувачу не потрібно вводити свої дані ще раз.

- *mainContent.guard* перевіряє валідність JWT-токену. Якщо він валідний, користувач допускається до сторінки з головним контентом, якщо ні – користувачу потрібно авторизуватися

- *admin.guard* дає дозвіл користувачу при наявності ролі адміністратора переходити до сторінок з пошуком будівлі та керування копіями бази даних.

Майже усі компоненти звертаються до серверу для отримання або оновлення певних даних. Так як усі запити до серверу повинні містити JWT-токен, то для спрощення відправки запиту був розроблений клас AddHeaderInterceptor.ts, який відповідає за автоматичне додавання заголовку авторизації до кожного запиту. Код цього класу наведений у додатку Е.

Для того, щоб відповідальна людина могла оповістити користувачей, що знаходяться у будівлі про небезпечну ситуацію, на сервері використовуються web-sockets. Для використання цієї технології у Angular використовується бібліотека @aspnet/signalr

Для стилізації деяких елементів використовується css framework – bootstrap. Для інтернаціоналізації інтерфейсу було застосовано пакет @angular/localize.

Створені компоненти розподілені по пакетах. Для зображення цього розподілу була створена діаграма пакетів (див. додаток В). На ній зображено те, за що відповідає окремий компонент у додатку. За генерування підвалу сайту та за інтерфейс авторизації/реєстрації відповідають пакети Login та Footer відповідно. Також у всьому проекті для перевірки не випадковості виконання зміни даних, наприклад видалення приміщення, використовується спеціальне діалогове вікно. За генерування цього вікна відповідає пакет ConfirmDialog. Пакет MainContent містить компоненти що відповідають за виконання базових дія для зареєстрованого користувача – зміна персональних даних (компонент Account), навігація по сайту (компонент MainContentHeader). Для дій адміністратора розроблений окремий підпакетAdmin. Він містить компонент для відображення дій адміністратора – створення резервної копії бази даних (компонент AdminPanel), список вже створених копій, які можна відновити або видалити (компонент BackupList) Підпакет Buildings містить елементи для пошуку будівлі, її зміни, а також має пакет Compartment, який містить компоненти для редагування інформації про приміщення.

Для опису дій, які потрібно виконати, щоб змінити інформацію про приміщення була розроблена діаграма взаємодій (див. додаток Г). Вона описує процес того, які кроки потрібно виконати, щоб внести нову інформацію про будівлю. До зміни будівлі також належать дії по зміні інформації внутрішньої структури будівлі такої як інформація про кімнати, поверхи, IoT пристроїв, що використовуються у приміщень, та встановлення перевірочних тестів для користувачів.

Сайт містить наступні сторінки

- Сторінка реєстрації/авторизації користувача

- сторінка зміни персональної інформації
- сторінка пошуку будівлі за ідентифікатором або адресою
- сторінка з відображенням інформації про будівлю
- сторінка з відображенням інформації про приміщення (загальна інформація про приміщення (пристрої, точки масштабу, маршрут евакуації) та мапа самого приміщення)
- окреме вікно для зміни інформації про тест, додавання інформації про пристрій, відповідальну людину для будівлі
- сторінка адміністратора, де можна створити копію бази даних або відновити з інших копій.

Кожна сторінка сайту підтримує локалізацію та інтернаціоналізацію. Це зроблено шляхом створення окремого файлу messages.uk.xlf, який використовує бібліотека @angular/localize. За перемикання мови відповідає кнопки Eng та Uk у верхньому правому куті.

Висновки: під час лабораторної роботи нами було розроблено клієнтську частину програмної системи для виявлення задимлення у приміщенні та допомоги при евакуації: спроектовано розмітку сторінок сайту, налаштована маршрутизація, забезпечений захист маршрутів, налаштовано отримання та відправлення інформації на сервер, організовано отримання повідомлень з серверу через використання технології SignalR, передбачено інтернаціоналізацію та локалізацію інтерфейсу сайту.

Додаток А

Діаграма прецедентів, що відображає відносини між акторами та прецедентами

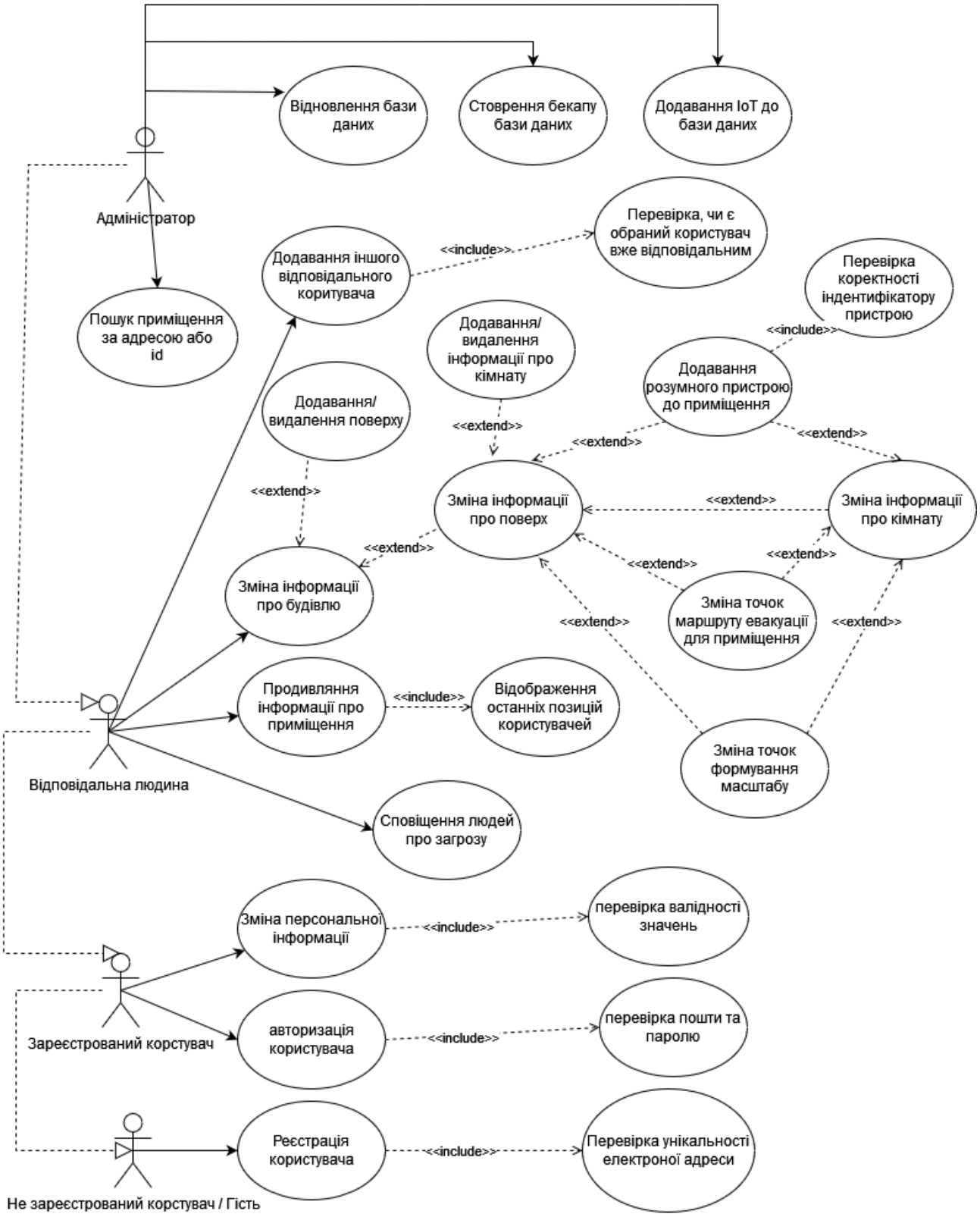

Рисунок А.1 – Use Case UML Diagram клієнтської частини FireSaver

Додаток Б

Діаграма компонентів – описує фізичну структура проєкту

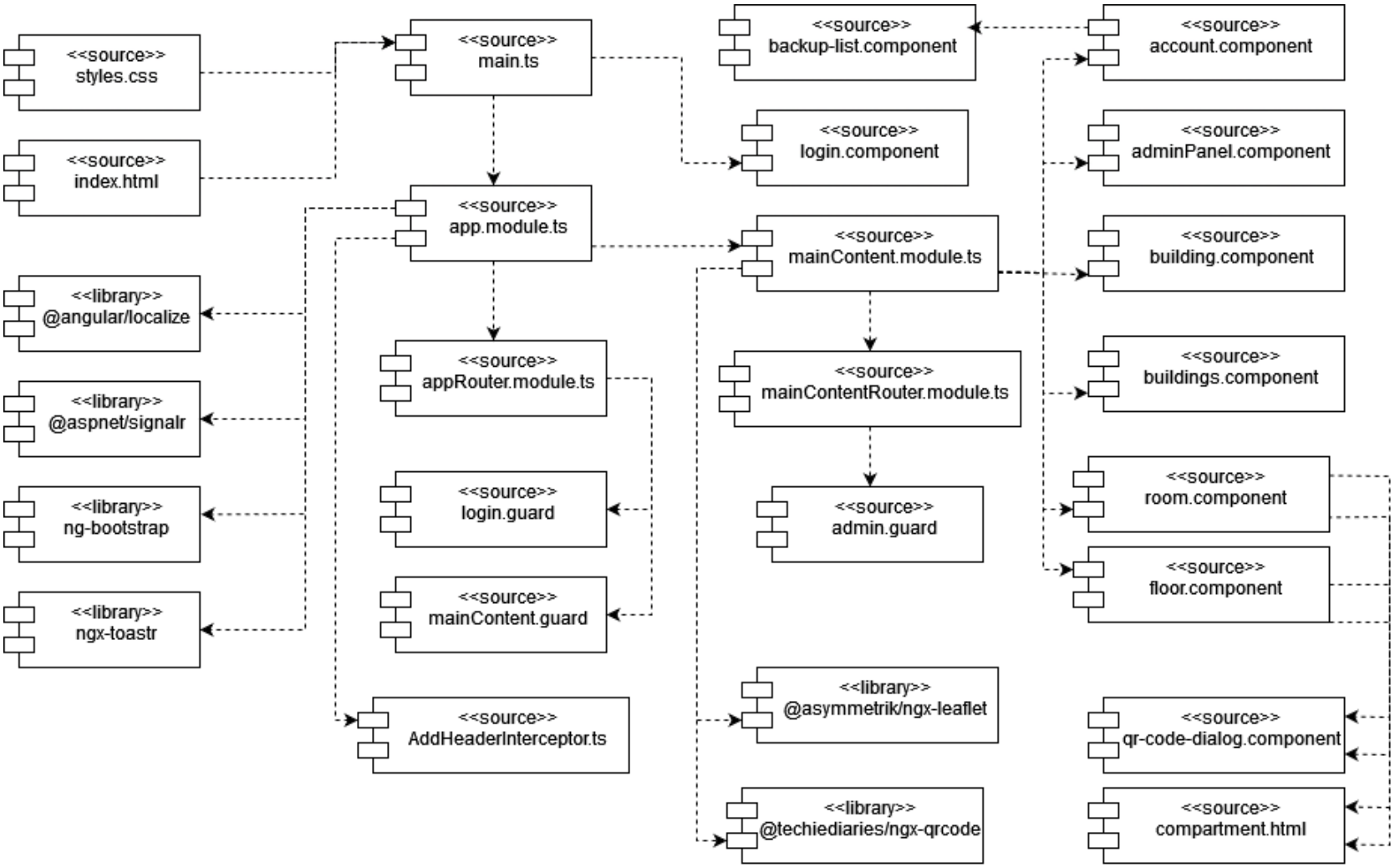

Рисунок Б.1 – Component Diagram клієнтської частини FireSaver

Додаток В

Діаграма пакетів зображує пакети и проекті та залежності між ними

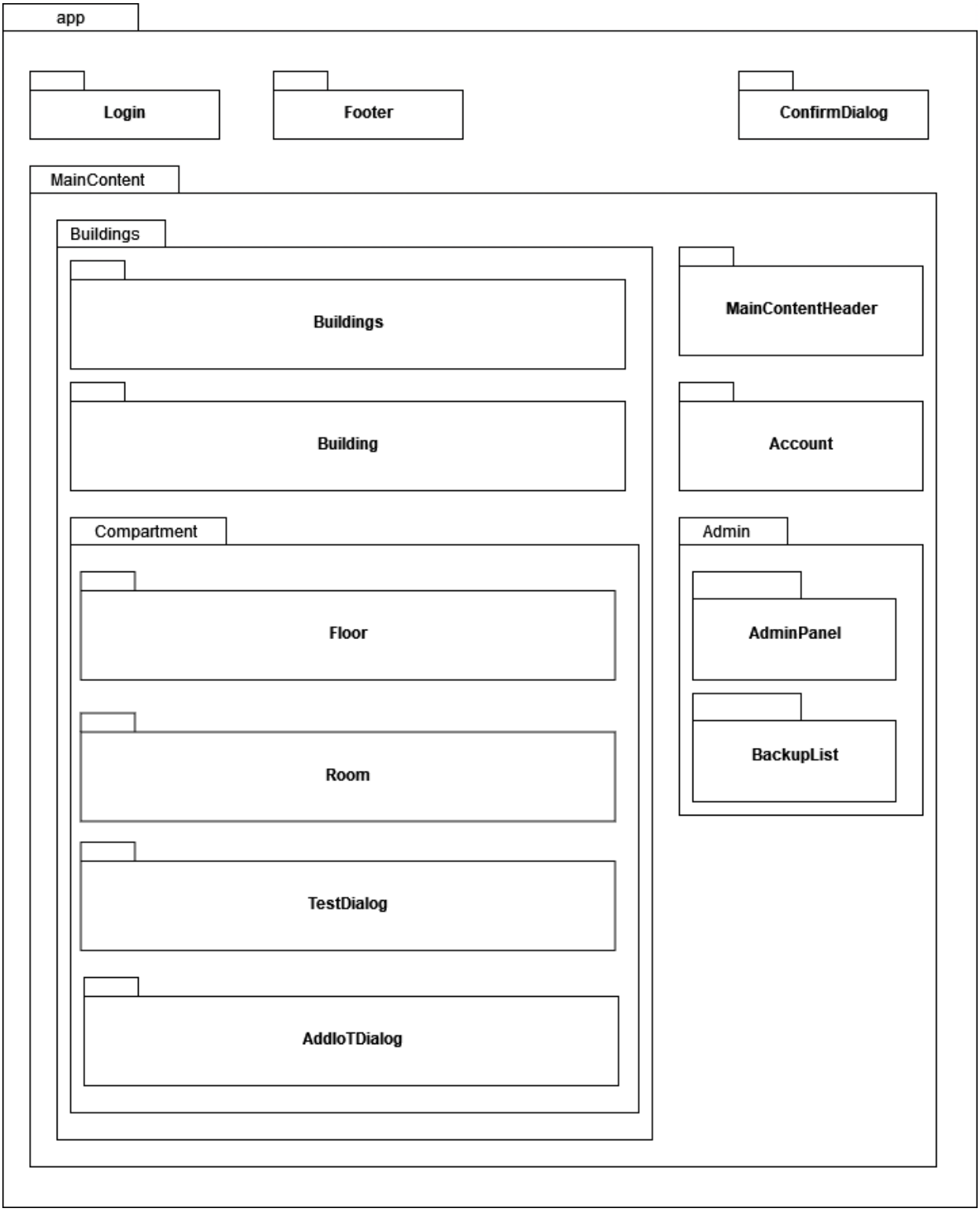

Рисунок В.1 – Package Diagram

Додаток Г

Діаграма взаємодії – зображує процес оновлення даних про будівлю

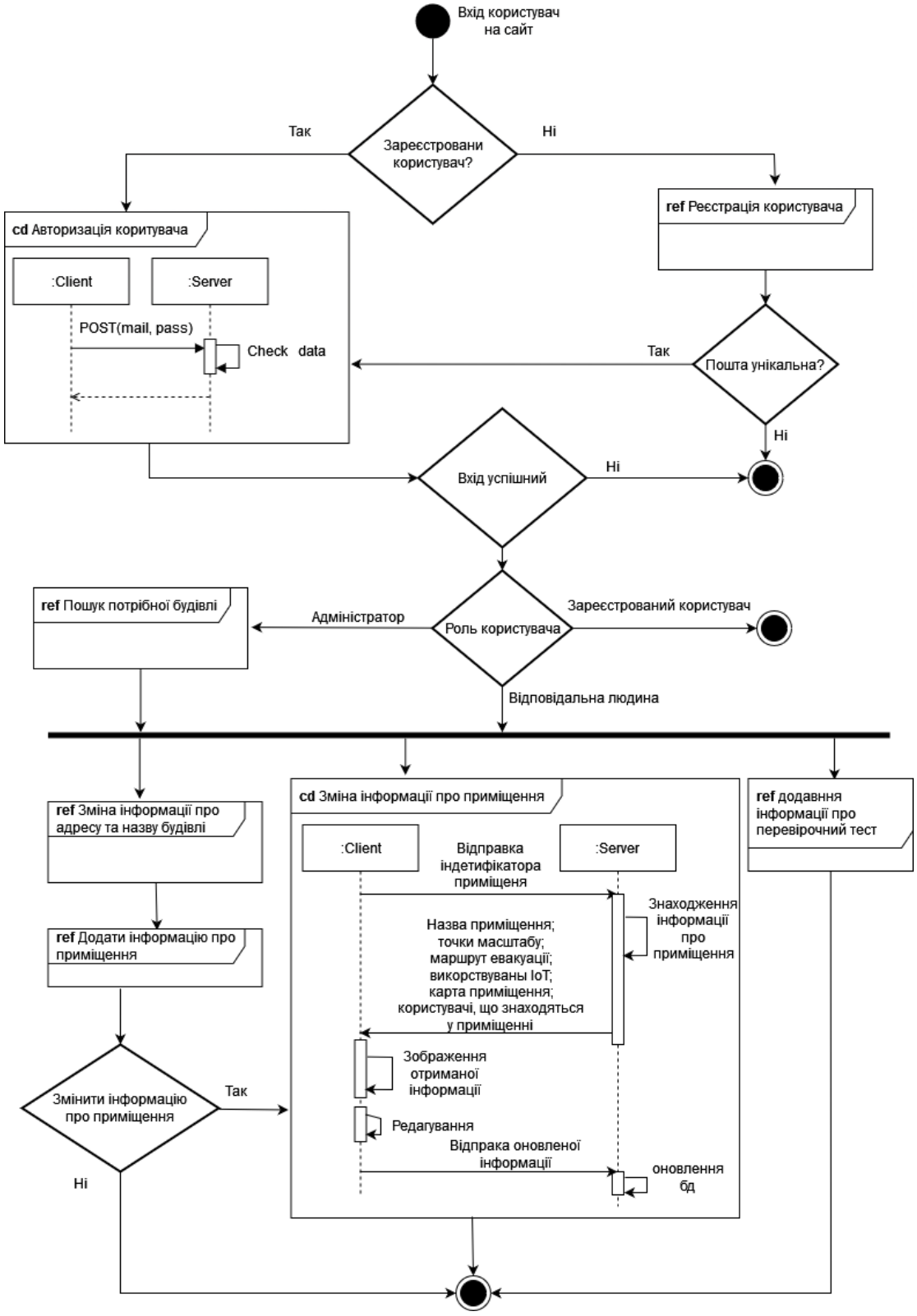

Рисунок Г.1 - Interaction Overview Diagram

Додаток Д

Фрагменти коду модулів, що відповідають за маршрутизацію на сайті

Модуль appRouter.module

```
@NgModule({
  imports: [
    RouterModule.forRoot([
      {
        path: 'entrance',
        component: LoginComponent,
        canActivate: [MainContentGuard]
      },
      {
        path: 'main', component: MainContentComponent,
        loadChildren: () => {
          return import('../../Modules/mainContent/mainContent.module')
        .then(m => m.MainContentModule)},
        canActivateChild: [LoginGuard]
      },
      { path: '', redirectTo: '/entrance', pathMatch: 'full' },
      { path: '**', redirectTo: '/entrance', pathMatch: 'full' },
    ])
  ],
  exports: [
    RouterModule
  ]
})
export class AppRouterModule { }

```

Модуль mainContentRouter.module

```
@NgModule({
  imports: [
    RouterModule.forChild([
      {
        path: 'room/:Id',
        component: RoomComponent
      },
      {
        path: 'floor/:Id',
        component: FloorComponent
      },
      {
        path: 'account',
        component: AccountComponent
      },
```

```
        {
          path: 'buildings',
          component: BuildingsComponent
        },
        {
          path: 'buildings/:buildingId',
          component: BuildingComponent
        },
        {
          path: 'adminPanel',
          component: AdminPanelComponent,
          canActivate: [AdminGuard]
        },
        {
          path: '', redirectTo: 'account',
          pathMatch: 'full'
        },
        {
          path: '**', redirectTo: 'account',
          pathMatch: 'full'
        }
      ])
    ],
    exports: [
      RouterModule
    ]
  })
  export class MainContentRouterModule { }
```

Додаток Е

Код класа AddHeaderInterceptor.ts, що відповідає за додавання токену авторизації

```
export class AddHeaderInterceptor implements HttpInterceptor {
    intercept(req: HttpRequest<any>, next: HttpHandler):
                                                Observable<HttpEvent<any>> {
        const bearerToken = localStorage.getItem('token');
        console.log(req);
        if (bearerToken) {
            const clonedRequest = req.clone({
                headers: req.headers.append('Authorization', 'Bearer ' +
                                                                bearerToken)
            });
            return next.handle(clonedRequest);
        }
        else {
            return next.handle(req);
        }
    }
}
```