Inżynieria Oprogramowania

Sprawozdanie z laboratorium

Projekt Hallowed Connection

Aplikacja do wyszukiwania punktów z dostępem do energii elektrycznej

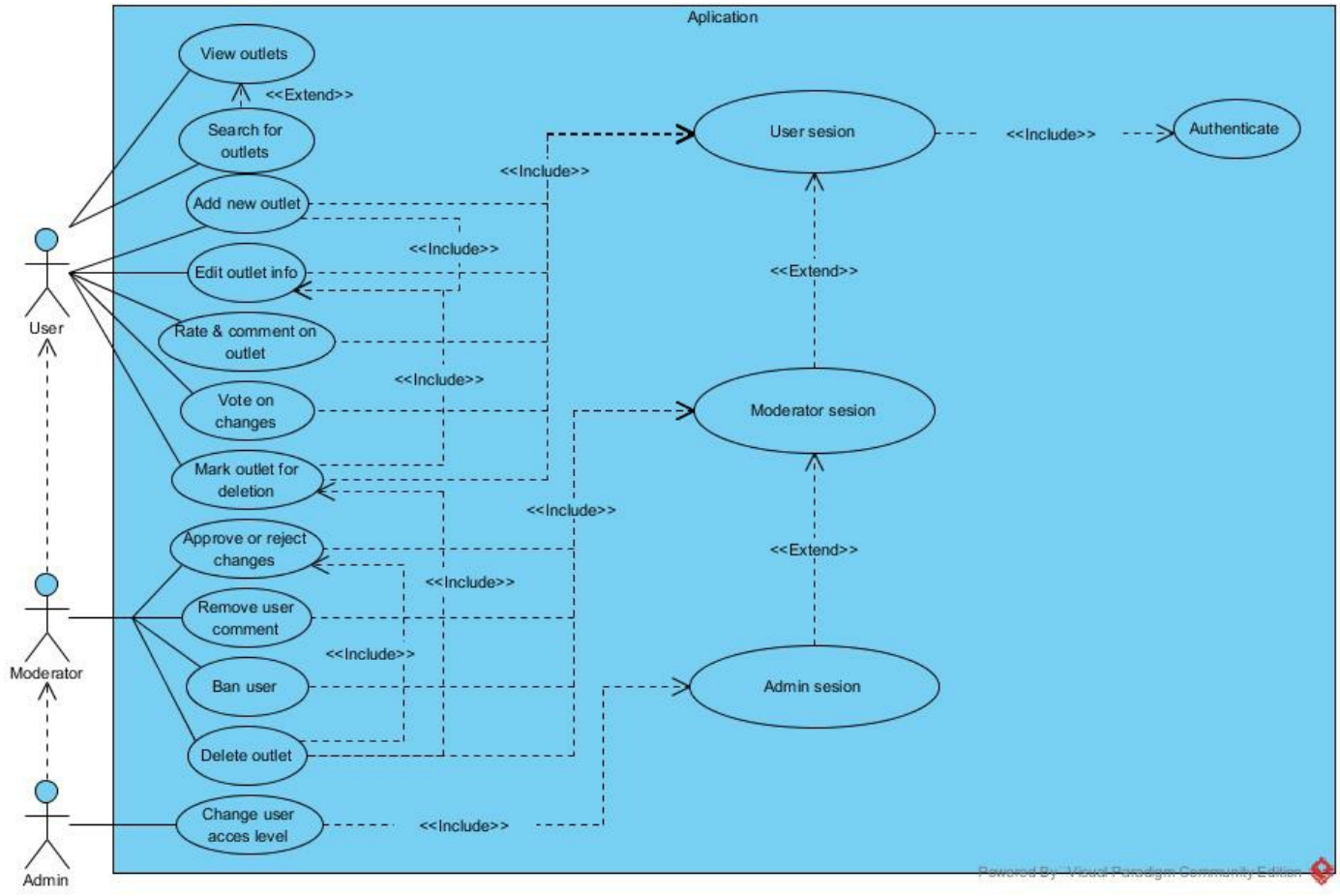
Uczestnicy	Prowadzący	Termin zajęć
Szymon Abramski Iwo Bujkiewicz	Dr inż. Roman Ptak	Środa 9:15

Zadanie 2

Celem laboratorium drugiego było stowrzenie modelu świata biznesowego i rzeczywistego, obecnego przy tworzenie aplikacji. Model miał być stworzony w dowolnym edytorze tekstu i jako jedyne zadanie na laboratoriach nie zawierał diagramów.

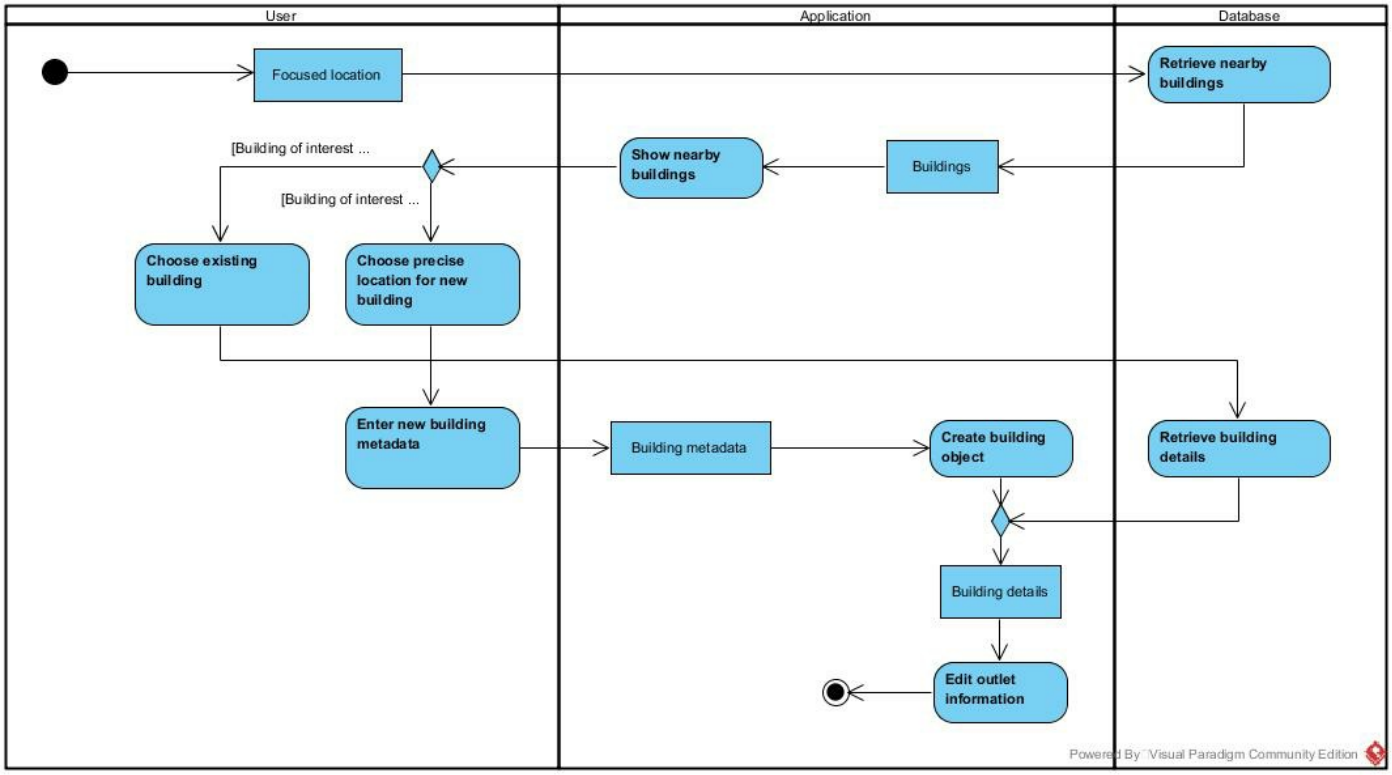
Zadanie 3

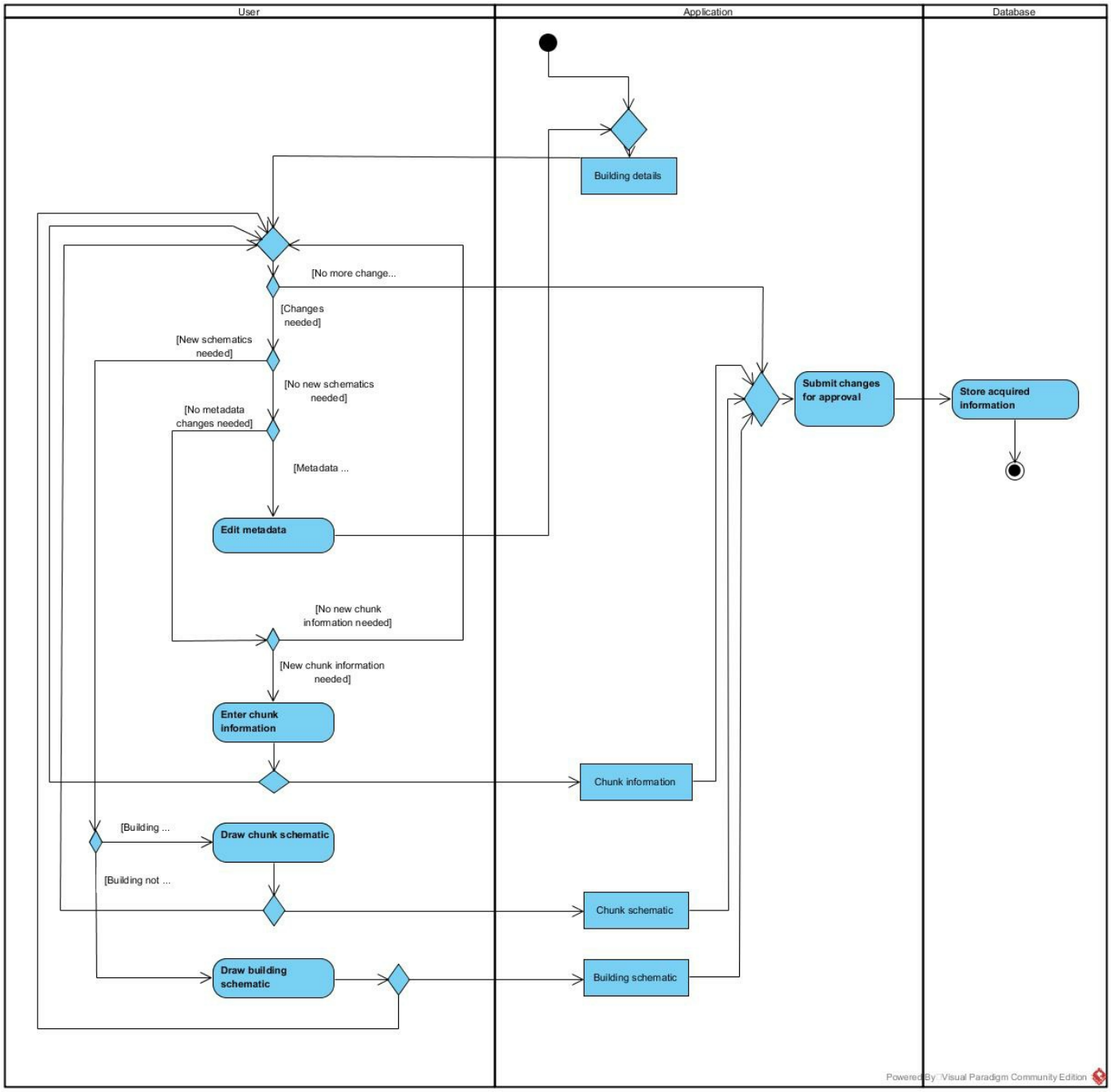
Celem laboratorium trzeciego było stworzenie diagramu przypadków użycia, opisującego działanie aplikacji. Diagram pokazuje wszystkie dostępne poziomy dostępu, wraz ze wszystkimi dostępnymi funkcjami aplikacji w zależności od poziomu. Każdy poziom dostępu jest ściśle powiązany z odpowiadającymi im sesjami. Rozróżniamy sesję anoniomwą, sesję użykownika, sesję administratora oraz sesję moderatora. Dla przykładu, administrator w odpowiadającej mu sesji może robić wszystko to co niższe poziomy zabezpieczeń, a oprócz tego nadawać poziomy dotępu innym użytkownikom.

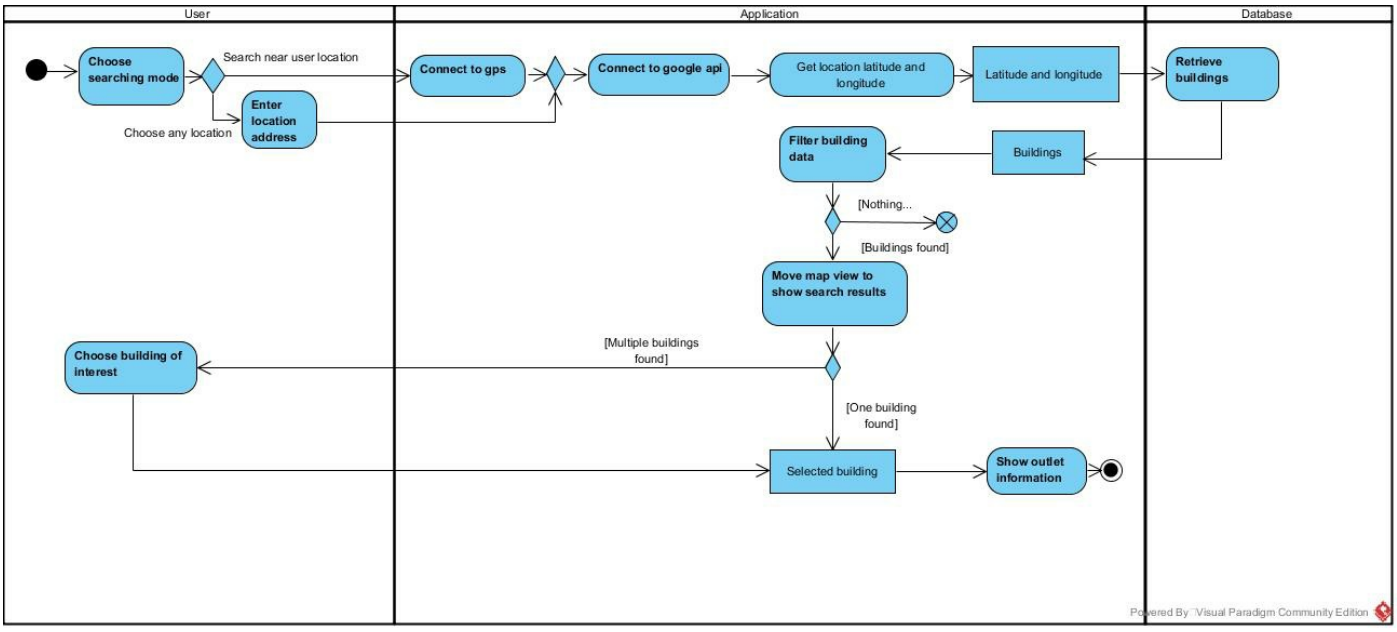


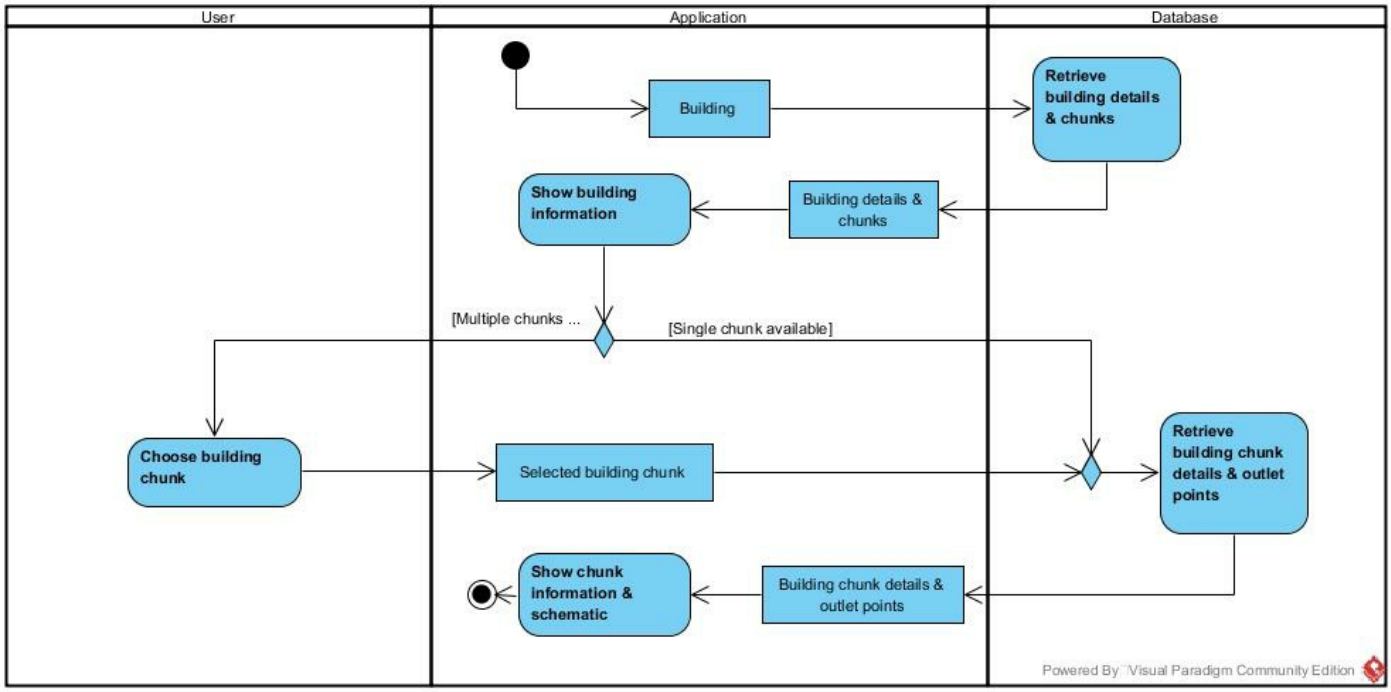
Zadanie 4

Celem laboratorium czwartego była budowa diagramu czynności reprezentującego model biznesowy „świata rzeczywistego” na podstawie wykonanego opisu procesów biznesowych oraz budowa diagramów czynności reprezentujących scenariusze wybranych przypadków użycia. Każdy diagram powinien zawierać tory, w tym przypadku jest to użytkownik, aplikacja i baza danych. Każdy diagram opisuje drogę przez tory jaką przebywa dana funkcja. Na torze użytownika, może on dokonowyać wyborów, za pomocą aplikacji, które wpływają na zakończenie przebiegu funkcji. Następnie aplikacja łączy się z bazą danych, z której pobiera informaje żądane przez użytkownika.



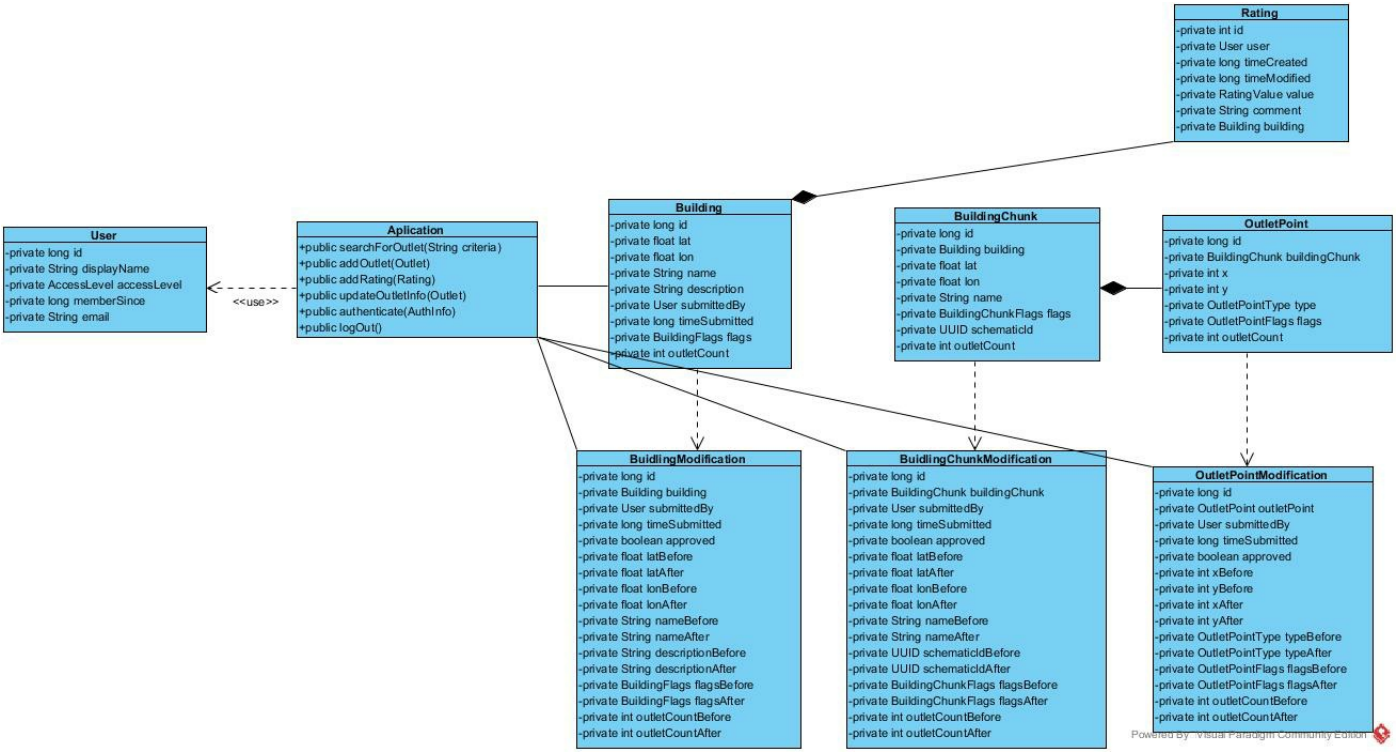

User
Application
Database
Building details
[No more change...
[Changes needed]
[New schematics needed]
[No new schematics needed]
[No metadata changes needed]
[Metadata ...
Edit metadata
Submit changes for approval
Store acquired information
[No new chunk information needed]
[New chunk information needed]
Enter chunk information
Chunk information
[Building ...
Draw chunk schematic
[Building not ...
Chunk schematic
Draw building schematic
Building schematic
Powered By Visual Paradigm Community Edition


User
Application
Database
Choose searching mode
Search near user location
Choose any location
Enter location address
Connect to gps
Connect to google api
Get location latitude and longitude
Latitude and longitude
Retrieve buildings
Buildings
Filter building data
[Nothing...
[Buildings found]
Move map view to show search results
[Multiple buildings found]
Choose building of interest
[One building found]
Selected building
Show outlet information
Powered By Visual Paradigm Community Edition



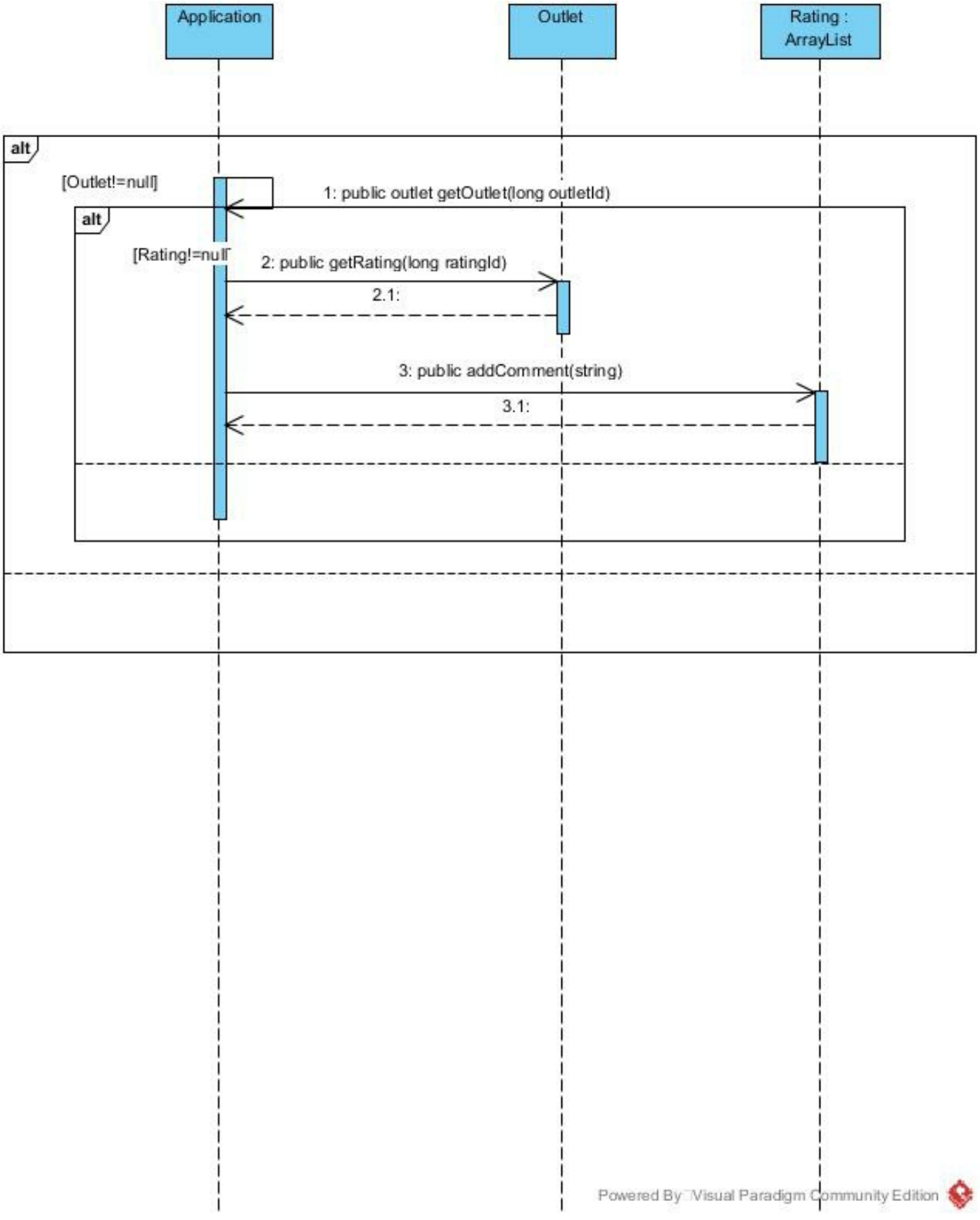
Zadanie 5

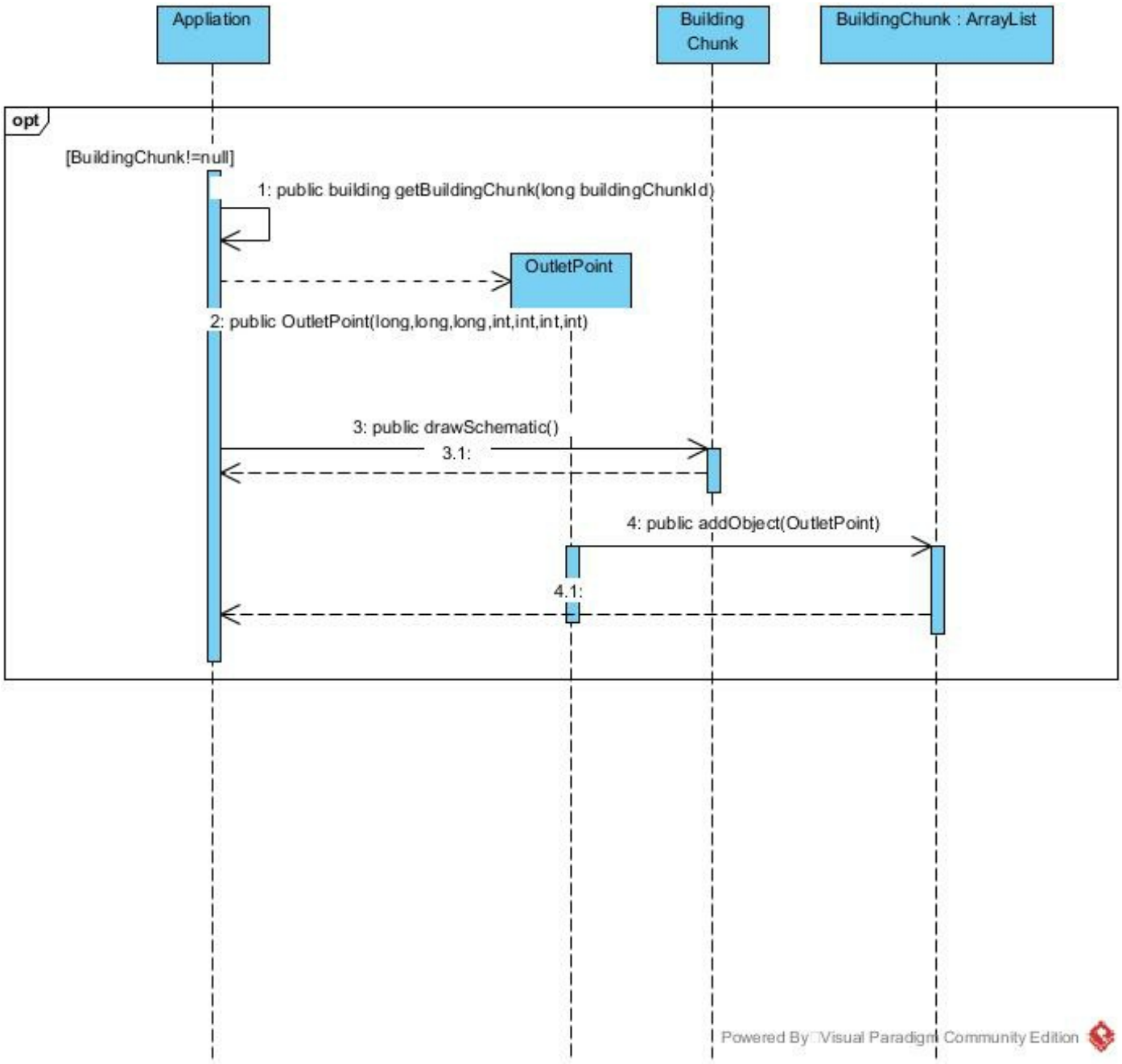
Celem laboratorium piątego była identyfikacja klas reprezentujących logikę biznesową projektowanego oprogramowania, definicja atrybutów i operacji klas oraz związków między klasami na podstawie analizy scenariuszy przypadków użycia. W diagrami klas opisane są wszystkie atrybuty i funkcje dostępne w klasie. Na diagramie przedstawiono również związki między klasami.

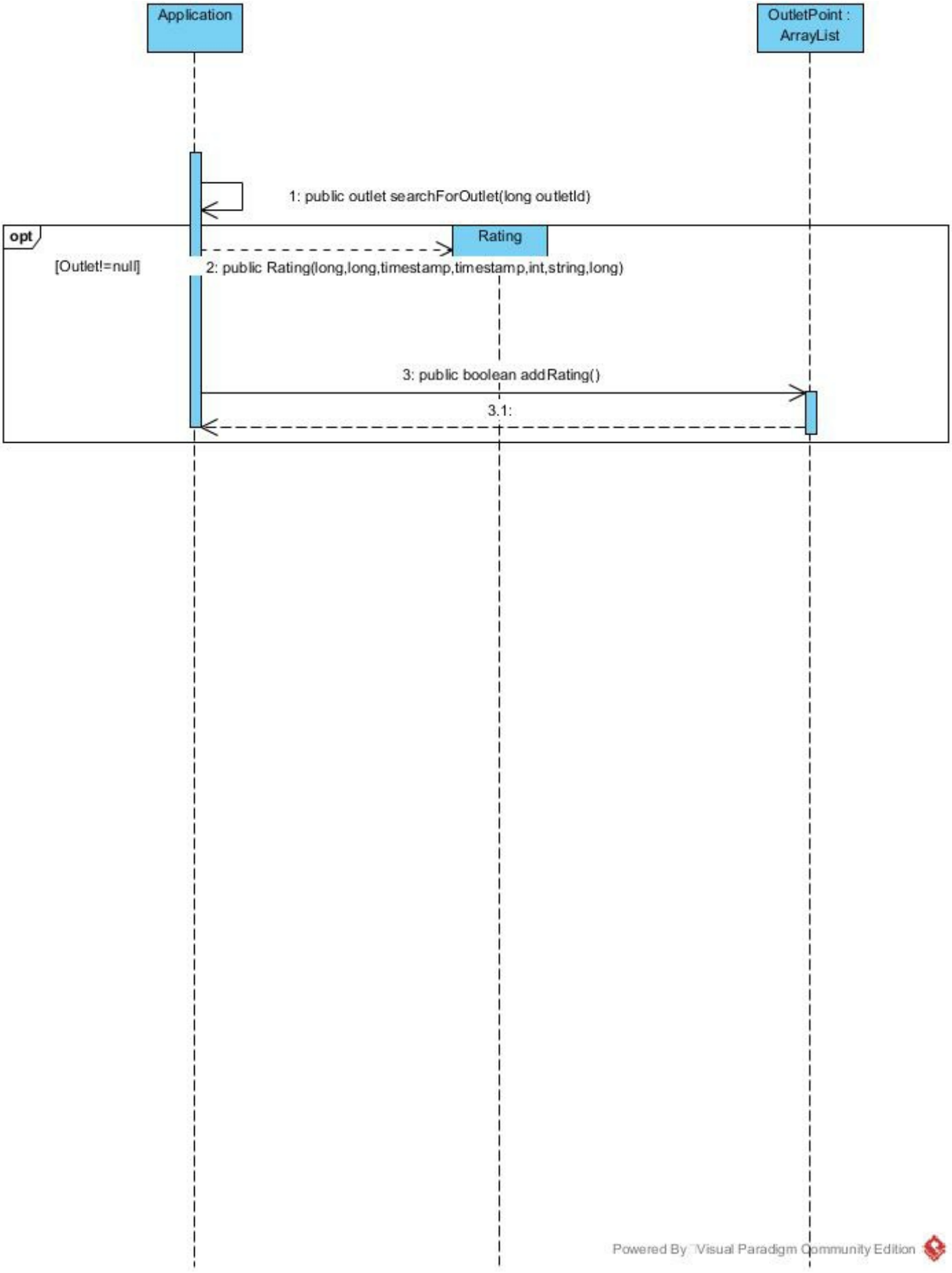


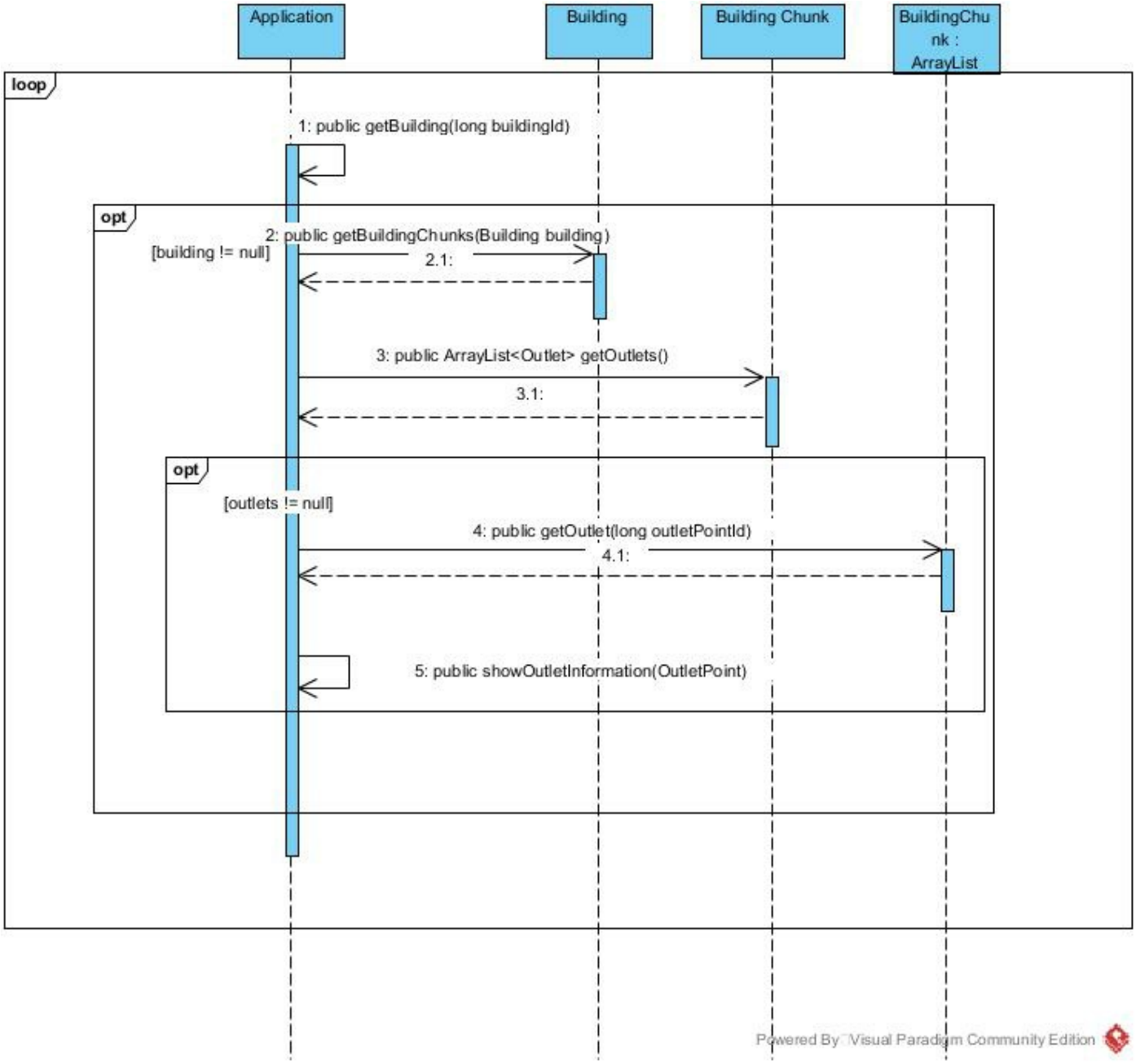
Zadanie 6

Celem laboratorium szóstego było opracowanie diagramów sekwencji dla wybranych przypadków użycia reprezentujących usługi oprogramowania wynikających również z wykonanych diagramów czynnosc.



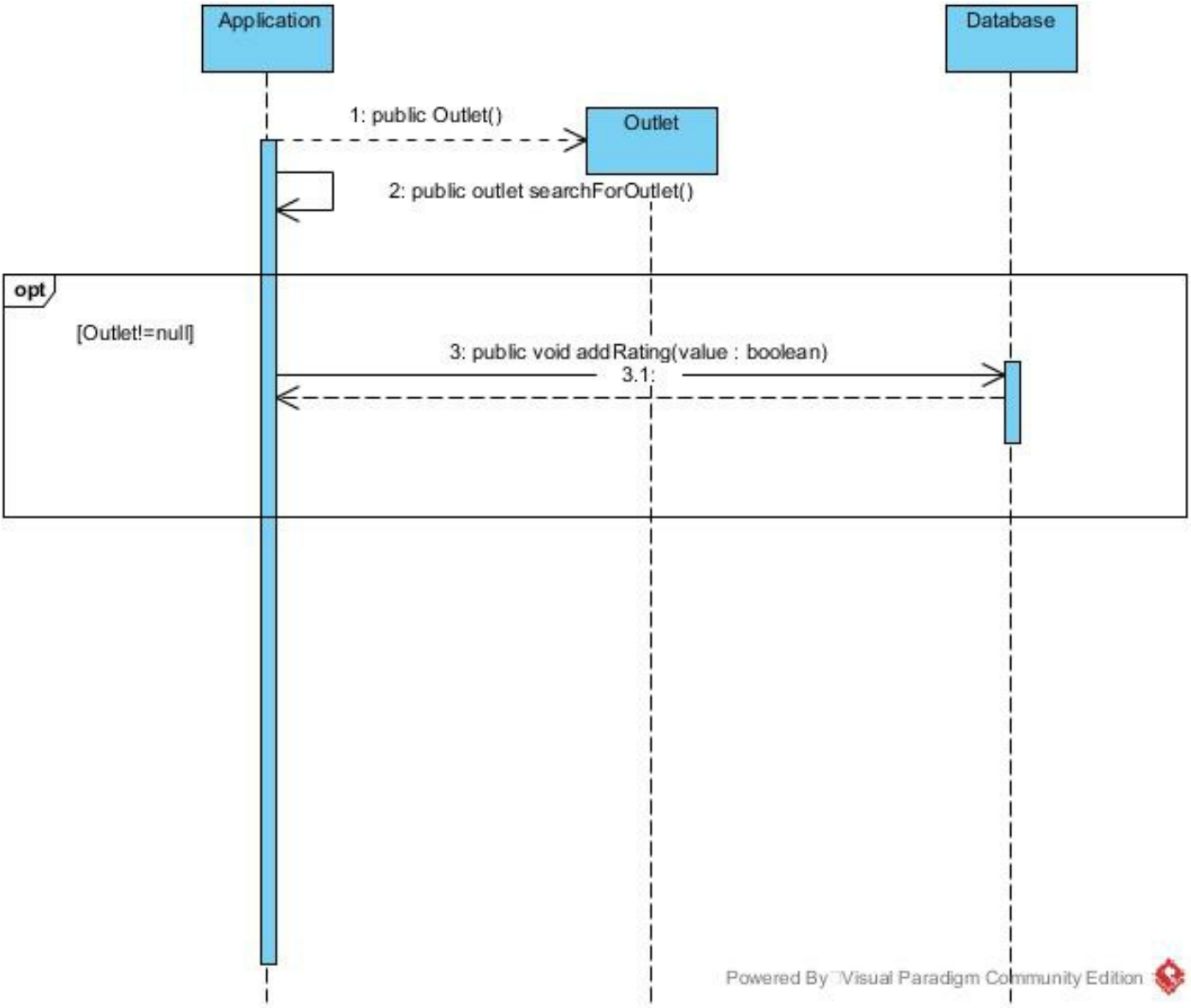

Appliation
Building Chunk
BuildingChunk : ArrayList
opt
[BuildingChunk!=null]
1: public building getBuildingChunk(long buildingChunkId)
OutletPoint
2: public OutletPoint(long,long,long,int,int,int,int)
3: public drawSchematic()
3.1:
4: public addObject(OutletPoint)
4.1:
Powered By Visual Paradigm Community Edition

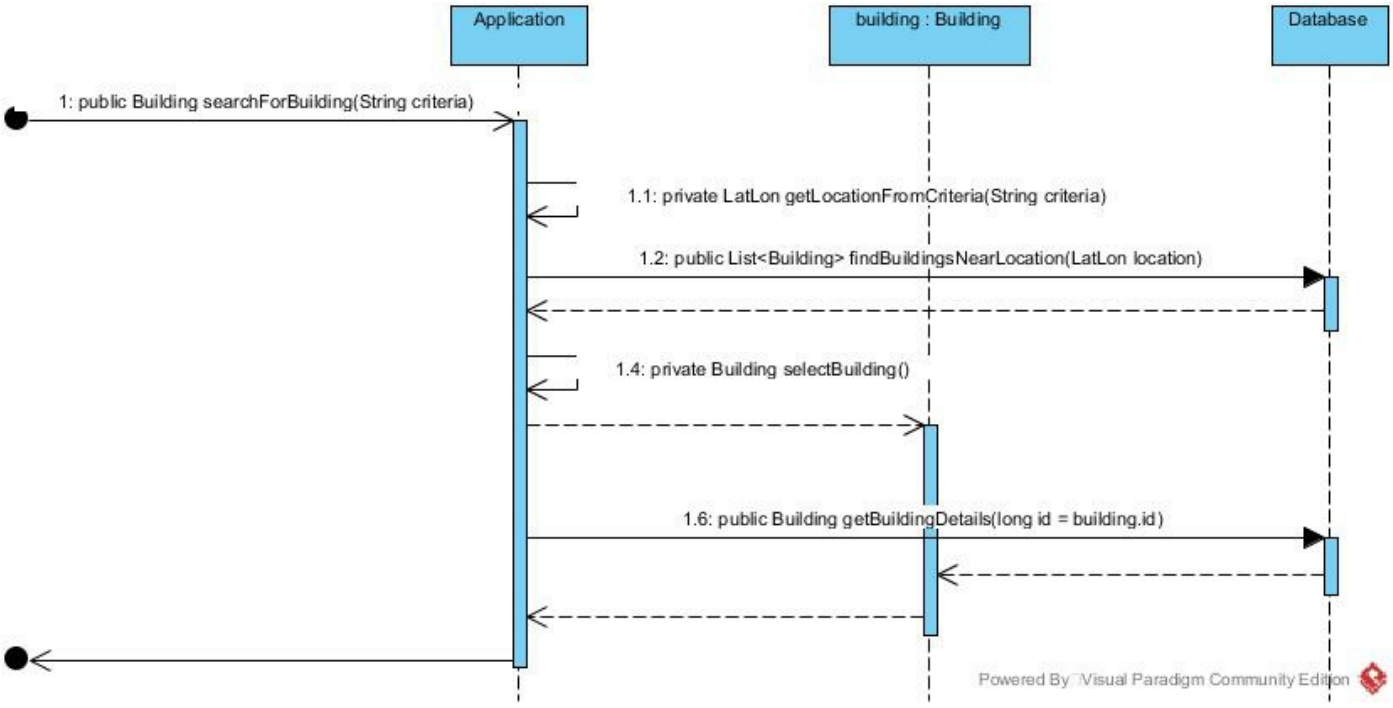



Application
Building
Building Chunk
BuildingChunk : ArrayList
loop
1: public getBuilding(long buildingId)
opt
2: public getBuildingChunks(Building building)
[building != null]
2.1:
3: public ArrayList<Outlet> getOutlets()
3.1:
opt
[outlets != null]
4: public getOutlet(long outletPointId)
4.1:
5: public showOutletInformation(OutletPoint)
Powered By Visual Paradigm Community Edition

Zadanie 7

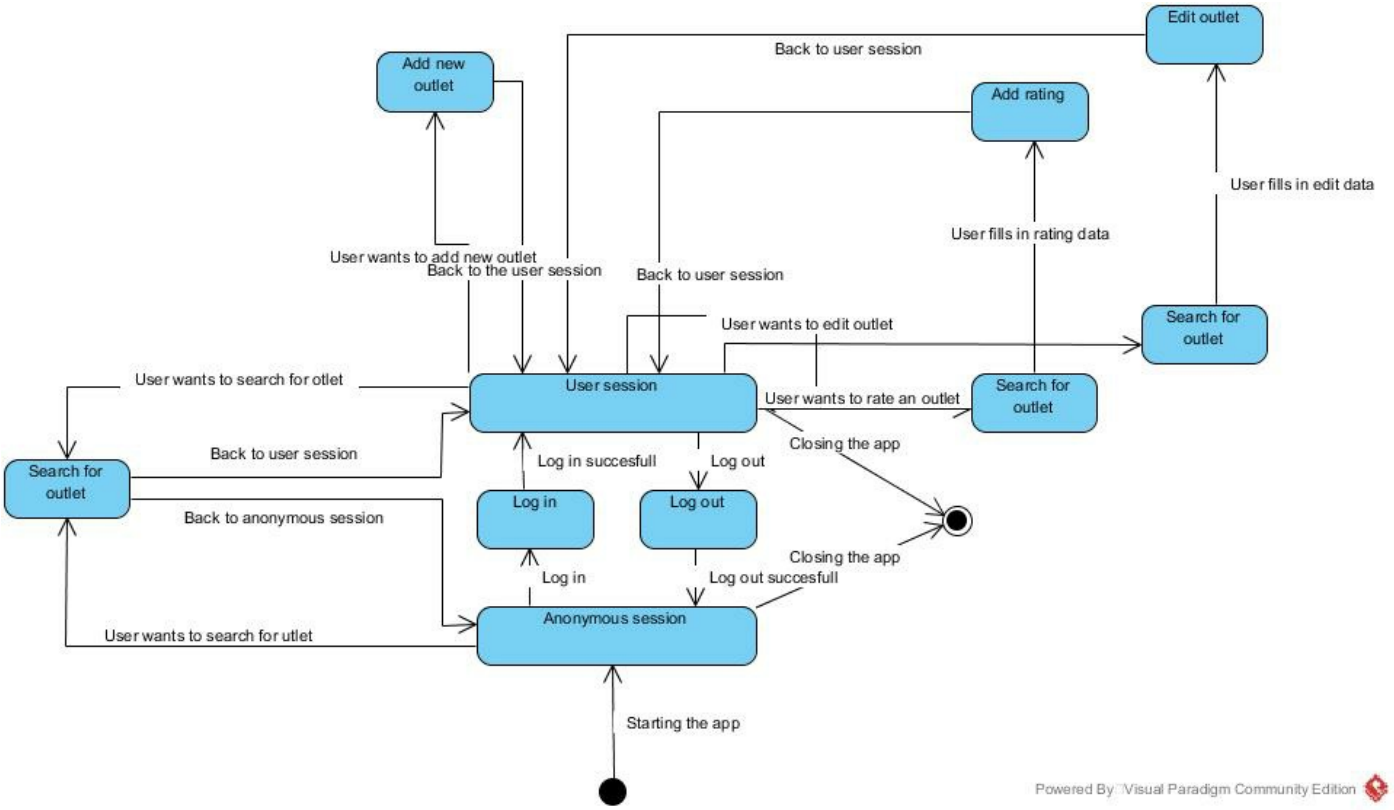
Celem laboratorium siódmego było opracowanie diagramów sekwencji dla wybranych przypadków użycia reprezentujących usługi oprogramowania wynikających również z wykonanych diagramów czynności.

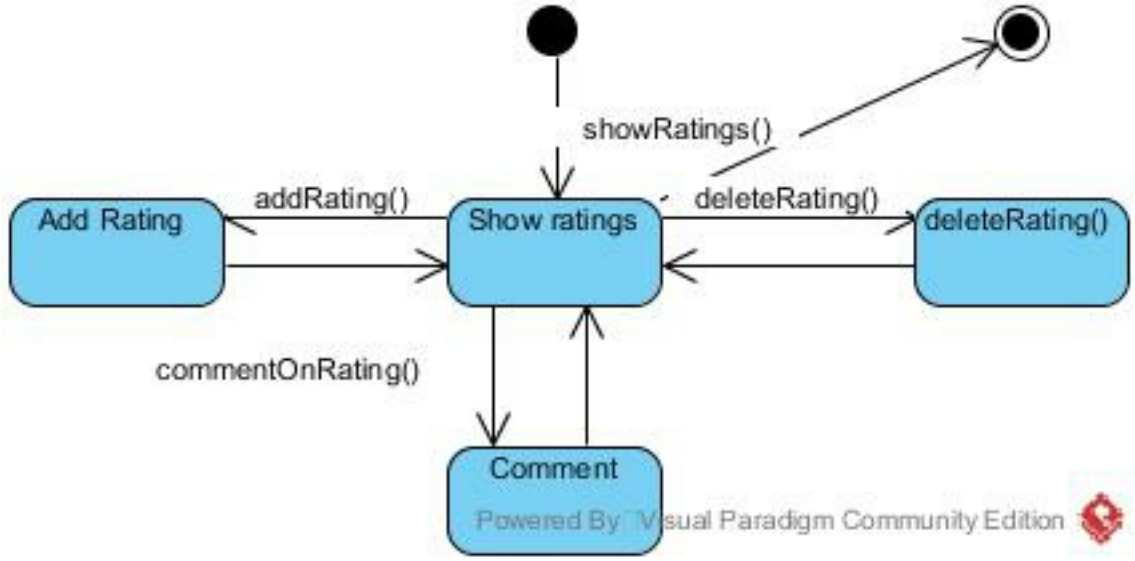




Zadanie 8

Celem laboratorium ósmego było opracowanie diagramu stanów dla wybranej klasy, reprezentującego wpływ różnych przypadków użycia na zmiany stanów tej klasy, modelowanych za pomocą diagramów sekwencji.





Zadanie 9

Celem laboratorium dziewiątego było napisanie zestawu testów jednostkowych dla projektowanej aplikacji. Opracowano testy jednostkowe testujące przede wszystkim zachowanie klasy reprezentującej logikę dostępu do bazy danych, ale także zachowanie klas encyjnych. Poniżej zaprezentowano kilka przykładowych testów.

```
//  class OutletInfoManagerTest

    @Test
    void addNewBuilding() throws SQLException {
        BuildingModification buildingModification = new BuildingModification();
        long randomId = random.nextLong();
        buildingModification.setId(randomId);

buildingModification.setSubmittedBy(outletInfoManager.getUser(-19248712047002142L));
        buildingModification.setTimeSubmitted(Timestamp.from(Instant.now()));
        buildingModification.setApproved(true);
        buildingModification.setLatAfter(51.2137);
        buildingModification.setLonAfter(17.1410);
        buildingModification.setNameAfter("JUnitTest");
        buildingModification.setDescriptionAfter(Integer.toString(random.nextInt()));
        buildingModification.setFlagsAfter(BuildingFlags.NONE);

        outletInfoManager.addNewBuilding(buildingModification);

        newBuildingId =
outletInfoManager.getBuildingModification(randomId).getBuilding().getId();
    }
```

```
//  class UserTest

    @Test
    void getMemberSince() throws SQLException {
        User user = outletInfoManager.getUser(76547231626540123 54L);
        assertEquals(Timestamp.valueOf("2018-01-21 20:50:37.61357"),
user.getMemberSince());
    }
```

```
//  class BuildingTest

    @Test
    void getName() throws SQLException {
        Building building = outletInfoManager.getBuilding(720195837538547236L);
        assertEquals("Pasaż Grunwaldzki", building.getName());
    }
```

```
//  class BuildingModificationTest

    @Test
    void getFlagsAfter() throws SQLException {
        BuildingModification buildingModification =
outletInfoManager.getBuildingModification(9847211560751L);
        assertEquals(BuildingFlags.NONE, buildingModification.getFlagsAfter());
    }
```

Zadanie 10

Celem laboratorium dziesiątego było utworzenie testów akceptacyjnych dla projektowanej aplikacji przy użyciu narzędzia FitNesse. Poniżej zaprezentowano przykładowy test w narzędziu FitNesse.

```
!define TEST_SYSTEM {slim}
!path /home/outfrost/git/hallowed-
connection/app/server/fitnesse/FitNesseRoot/target/test-classes
!path /home/outfrost/git/hallowed-
connection/app/server/fitnesse/FitNesseRoot/target/classes
!path /home/outfrost/git/hallowed-
connection/app/server/fitnesse/FitNesseRoot/target/libraries

!|import|
|org.postgresql.Driver|
|java.sql.*|
|java.time.LocalDateTime|
|java.util.EnumSet|
|java.util.Random|
|java.util.UUID|
|java.io.Serializable|

!|outfrost.hallowedconnection.server.OutletInfoManagerFixture|
|getUserValidation?|
|OK|
```

```
    public void execute() throws SQLException, ClassNotFoundException {
        outletInfoManager = new OutletInfoManager();

        User user = outletInfoManager.getUser(28464942894168541L);
        getUserValidation = (user.getDisplayName().equals("mkay") &&
user.getAccessLevel() == UserAccessLevel.REGISTERED &&
user.getEmail().equals("idontknow@thispersonsemail.com")) ? "OK" : "FAIL";

    }
```

FitNesse Test Edit Add Tools Execution Log

FrontPage / OutletInfoManagerFixture

Test Pages: 1 right, 0 wrong, 0 ignored, 0 exceptions **Assertions:** 1 right, 0 wrong, 0 ignored, 0 exceptions (0.610 seconds)

Test System: slim:fitnesse.slim.SlimService

variable defined: TEST_SYSTEM=slim
classpath: /home/outfrost/git/hallowed-connection/app/server/fitnesse/FitNesseRoot/target/test-classes
classpath: /home/outfrost/git/hallowed-connection/app/server/fitnesse/FitNesseRoot/target/classes
classpath: /home/outfrost/git/hallowed-connection/app/server/fitnesse/FitNesseRoot/target/libraries

import
org.postgresql.Driver
java.sql.*
java.time.LocalDateTime
java.util.EnumSet
java.util.Random
java.util.UUID
java.io.Serializable

outfrost.hallowedconnection.server.OutletInfoManagerFixture
getUserValidation?
OK