Base - система, що пов’язує інші підсистеми, шляхом передачі запитів від одних підсистем до інших.

Роль – зв’язок між іншими підсистемами.

**Юзкейси підсистеми Base:**

**Система доступу до аудиторії**

Діючі особи:  
 1. Reader

2. Handler

3. Maglock

Мета: отримати допуск до аудиторії.

Основний сценарій:

1. Reader зчитує UUID
2. Reader пересилає UUID на Handler
3. Handler відчиняє Maglock
4. Maglock відправляє сигнал про здійснення відкриття
5. На GUI пересилається повідомлення про успішне виконання роботи

Альтернативний сценарій:

3.1 Номер не є валідним

3.2 На GUI пересилається повідомлення про заборону доступу

Код у PlantUML:

@startuml Pass to Auditory

activate Reader

ref over Reader : Read UUID

Reader -> Handler : Send UUID

destroy Reader

activate Handler

ref over Handler : PAA: "Verify UUID"

alt gruanted

Handler -> Maglock : Open

activate Maglock

Maglock --> Handler : true

destroy Maglock

ref over Handler : GUI: Show c\_message(\n"Please enter")\nLog: maglock opened\nwith UUID

else denied

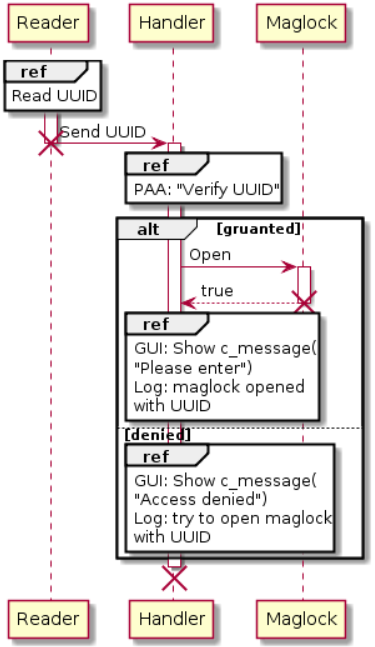
ref over Handler : GUI: Show c\_message(\n"Access denied")\nLog: try to open maglock\nwith UUID

end

destroy Handler

@enduml

Діаграма у PlantUML:



**Передавання розкладу системі**

Діючі особи:

1. Sender
2. Base

Мета: отримання розкладу

Основний сценарій:

1. Sender зчитує данні

2. Sender відправляє інформацію Base

3. DataBase надає необхідну інформацію

4. Logsystem проводить запис у logfile

5. Base відправляє розклад Sender

Альтернативний сценарій:

3.1 Logsystem проводить запис у logfile про помилку

3.2 Base відправляє повідомлення про помилку Sender

Код у PlantUML:

@startuml

ref over Sender: data acquisition

"Sender" -> "Base": send data

activate Base

alt performans

ref over Base: DataBase: receiving\_information()

ref over Base: Logsystem: write\_logfile()

"Base" --> "Sender": send schedule

else failure

ref over Base: Logsystem: write\_error\_logfile()

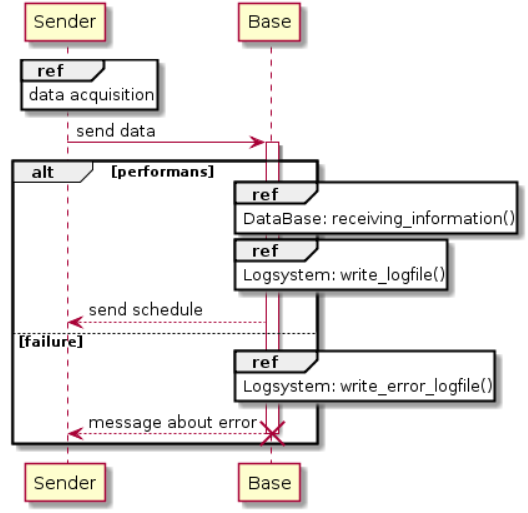
"Base" --> "Sender": message about error

destroy Base

end

@enduml

Діаграма у PlantUML:



**Надання права доступу**

Діючі особи:

1. Sender
2. Base

Мета: отримання права доступу

Основний сценарій:

1. Sender зчитує данні

2. Sender відправляє інформацію Base

3. DataBase надає право доступу

4. Logsystem проводить запис у logfile

5. Base надає права доступу Sender

Альтернативний сценарій:

3.1 Logsystem проводить запис у logfile про помилку

3.2 Base відправляє повідомлення про помилку Sender

Код у PlantUML:

@startuml

ref over Sender: data acquisition

"Sender" -> "Base": send data

activate Base

ref over Base: DataBase: add\_new\_law()

alt performans

ref over Base: Logsystem: write\_logfile()

"Base" --> "Sender": send law

else failure

ref over Base: Logsystem: write\_error\_logfile()

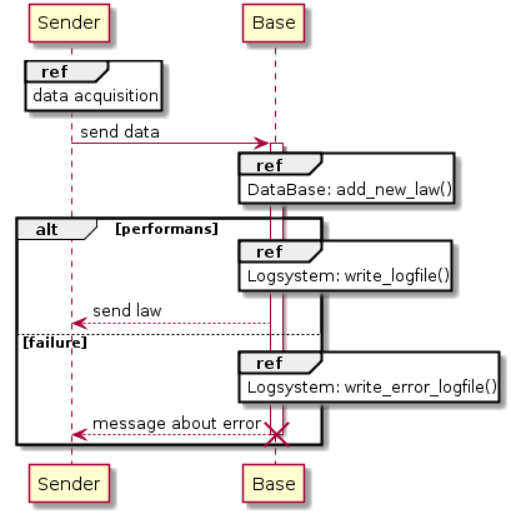
"Base" --> "Sender": message about error

destroy Base

end

@enduml

Діаграма у PlantUML:



**Додання викладача до системи**

Діючі особи:

1. Sender
2. Base

Мета: додання викладача до системи

Основний сценарій:

1. Sender зчитує данні

2. Sender відправляє інформацію Base

3. DataBase записує нового викладача до системи

4. Logsystem проводить запис у logfile

5. Base відправляє оновлений розклад Sender

Альтернативний сценарій:

3.1 Logsystem проводить запис у logfile про помилку

3.2 Base відправляє повідомлення про помилку Sender

Код у PlantUML:

@startuml

ref over Sender: data acquisition

"Sender" -> "Base": send request

activate Base

ref over Base: DataBase: add\_new\_lecturer()

alt performans

ref over Base: Logsystem: write\_logfile()

"Base" --> "Sender": send updated schedule

else failure

ref over Base: Logsystem: write\_error\_logfile()

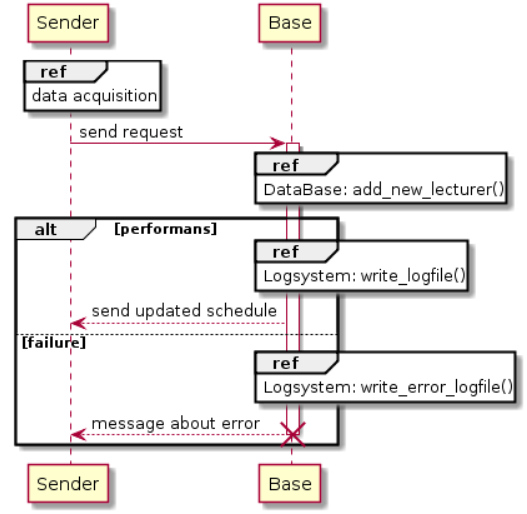
"Base" --> "Sender": message about error

destroy Base

end

@enduml

Діаграма у PlantUML:



**Зміна аудиторії**

Діючі особи:

1. Sender
2. Base

Мета: зміна інформації в системі щодо аудиторії

Основний сценарій:

1. Sender зчитує данні

2. Sender відправляє інформацію Base

3. DataBase замінює аудиторію

4. Logsystem проводить запис у logfile

5. Base відправляє оновлений розклад Sender

Альтернативний сценарій:

3.1 Logsystem проводить запис у logfile про помилку

3.2 Base відправляє повідомлення про помилку Sender

Код у PlantUML:

@startuml

ref over Sender: data acquisition

"Sender" -> "Base": send request

activate Base

ref over Base: DataBase: change\_audit()

alt performans

ref over Base: Logsystem: write\_logfile()

"Base" --> "Sender": send updated schedule

else failure

ref over Base: Logsystem: write\_error\_logfile()

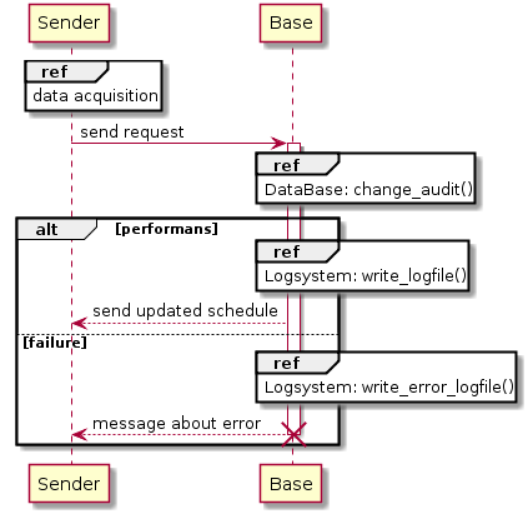
"Base" --> "Sender": message about error

destroy Base

end

@enduml

Діаграма у PlantUML:



**Видалення викладача з розкладу**

Діючі особи:

1. Sender
2. Base

Мета: видалити викладача з розкладу

Основний сценарій:

1. Sender зчитує данні

2. Sender відправляє інформацію Base

3. DataBase видаляє викладача з розкладу

4. Logsystem проводить запис у logfile

5. Base відправляє повідомлення про видалення Sender

Альтернативний сценарій:

3.1 Logsystem проводить запис у logfile про помилку

3.2 Base відправляє повідомлення про помилку Sender

Код у PlantUML:

@startuml

ref over Sender: data acquisition

"Sender" -> "Base": send request

activate Base

ref over Base: DataBase: remove\_lecturer()

alt performans

ref over Base: Logsystem: write\_logfile()

"Base" --> "Sender": message about remove

else failure

ref over Base: Logsystem: write\_error\_logfile()

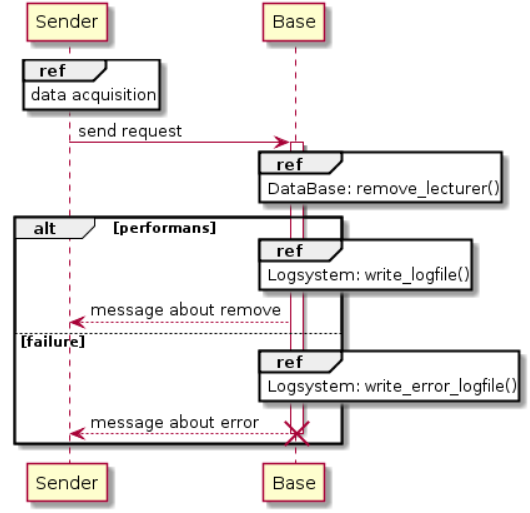
"Base" --> "Sender": message about error

destroy Base

end

@enduml

Діаграма у PlantUML:



**Додавання розкладу**

Діючі особи:

1. Sender
2. Base

Мета: додати нову інформацію щодо розкладу

Основний сценарій:

1. Sender зчитує данні

2. Sender відправляє інформацію Base

3. DataBase доповнює розклад новою інформацією

4. Logsystem проводить запис у logfile

5. Base відправляє повідомлення про доповнення розкладу Sender

Альтернативний сценарій:

3.1 Logsystem проводить запис у logfile про помилку

3.2 Base відправляє повідомлення про помилку Sender

Код у PlantUML:

@startuml

ref over Sender: data acquisition

"Sender" -> "Base": send request

activate Base

ref over Base: DataBase: add\_schedule()

alt performans

ref over Base: Logsystem: write\_logfile()

"Base" --> "Sender": message about \nadd new schedule

else failure

ref over Base: Logsystem: write\_error\_logfile()

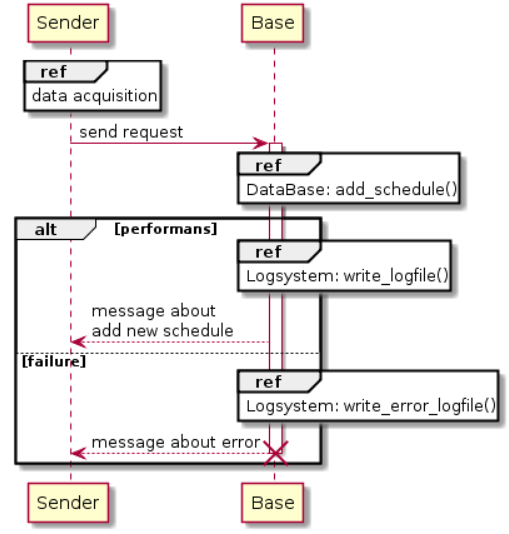
"Base" --> "Sender": message about error

destroy Base

end

@enduml

Діаграма у PlantUML:



**Видалення розкладу**

Діючі особи:

1. Sender
2. Base

Мета: видалити деяку інформацію щодо розкладу

Основний сценарій:

1. Sender зчитує данні

2. Sender відправляє інформацію Base

3. DataBase видаляє деяку частину розкладу

4. Logsystem проводить запис у logfile

5. Base відправляє повідомлення про видалення розкладу Sender

Альтернативний сценарій:

3.1 Logsystem проводить запис у logfile про помилку

3.2 Base відправляє повідомлення про помилку Sender

Код у PlantUML:

@startuml

ref over Sender: data acquisition

"Sender" -> "Base": send request

activate Base

ref over Base: DataBase: remove\_schedule()

alt performans

ref over Base: Logsystem: write\_logfile()

"Base" --> "Sender": message about \nremove schedule

else failure

ref over Base: Logsystem: write\_error\_logfile()

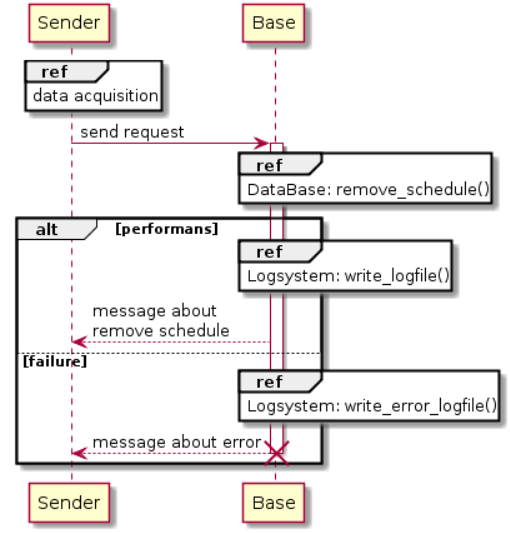
"Base" --> "Sender": message about error

destroy Base

end

@enduml

Діаграма у PlantUML:



**Доступ до обладнання**

Діючі особи:

1. Sender
2. Base

Мета: отримати доступ до обладнання

Основний сценарій:

1. Sender зчитує данні

2. Sender відправляє інформацію Base

3. DataBase видає дозвіл до використання обладнання

4. Logsystem проводить запис у logfile

5. Base відправляє дозвіл на використання обладнання Sender

Альтернативний сценарій:

3.1 Logsystem проводить запис у logfile про помилку

3.2 Base відправляє повідомлення про помилку Sender

Код у PlantUML:

@startuml

ref over Sender: data acquisition

"Sender" -> "Base": send request

activate Base

ref over Base: DataBase: permission\_access\_equipment()

alt performans

ref over Base: Logsystem: write\_logfile()

"Base" --> "Sender": message about \naccess to equipment

else failure

ref over Base: Logsystem: write\_error\_logfile()

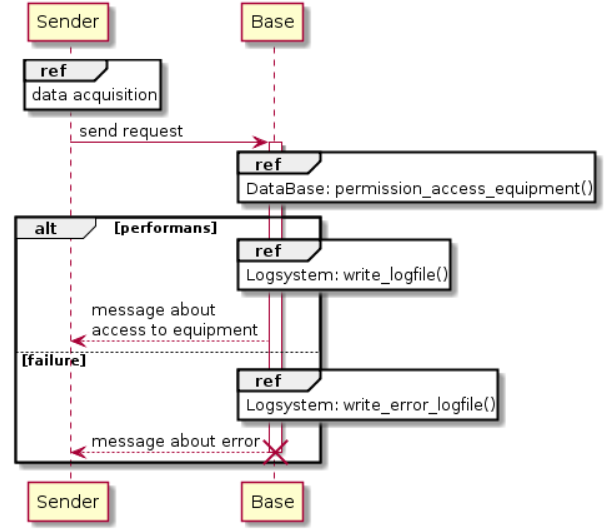
"Base" --> "Sender": message about error

destroy Base

end

@enduml

Діаграма у PlantUML:



**Повернення обладнання**

Діючі особи:

1. Sender
2. Base

Мета: оформлення повернення обладнання

Основний сценарій:

1. Sender зчитує данні

2. Sender відправляє інформацію Base

3. DataBase оформлює повернення обладнання

4. Logsystem проводить запис у logfile

5. Base відправляє дозвіл на повернення обладнання Sender

Альтернативний сценарій:

3.1 Logsystem проводить запис у logfile про помилку

3.2 Base відправляє повідомлення про помилку Sender

Код у PlantUML:

@startuml

ref over Sender: data acquisition

"Sender" -> "Base": send request

activate Base

ref over Base: DataBase: equipment\_return()

alt performans

ref over Base: Logsystem: write\_logfile()

"Base" --> "Sender": message about \nequipment return

else failure

ref over Base: Logsystem: write\_error\_logfile()

"Base" --> "Sender": message about error

destroy Base

end

@enduml

Діаграма у PlantUML:

