

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ
“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,
МЕХАНИКИ И ОПТИКИ”
ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2
«Построение диаграммы вариантов использования, диаграммы
классов»

Специальность 09.02.07 «Информационные системы и программирование»

ПМ.02 «Осуществление интеграции программных модулей»

МДК.2.1 «Технология разработки программного обеспечения»

Тема 2.1.2 «Описание и анализ требований. Диаграммы IDEF, DFD и UML»

Преподаватель: Коцюба И.Ю. «12» ноября 2025г. Оценка:	Выполнил: студент группы К3221 Доценников Н.А.
---	---

--	--

Санкт-Петербург

2025/2026

Цель работы.

Закрепить навыки моделирования предметной области с помощью UML: определить акторов и варианты использования, спроектировать логическую структуру классов и взаимосвязей для выбранной системы.

Задачи.

- Определить границы системы и ключевых заинтересованных лиц (акторов).
- Описать основные варианты использования, формализующие требования пользователя.
- Спроектировать диаграмму классов, отражающую доменную модель и сервисы.
- Обосновать выбор методологии управления разработкой (Scrum/Kanban/XP).

Индивидуальная тема.

Проектирование информационной системы «NoteManager» — кроссплатформенной системы заметок с единым упрощающим интерфейсом доступа к функциям (паттерн «Фасад»).

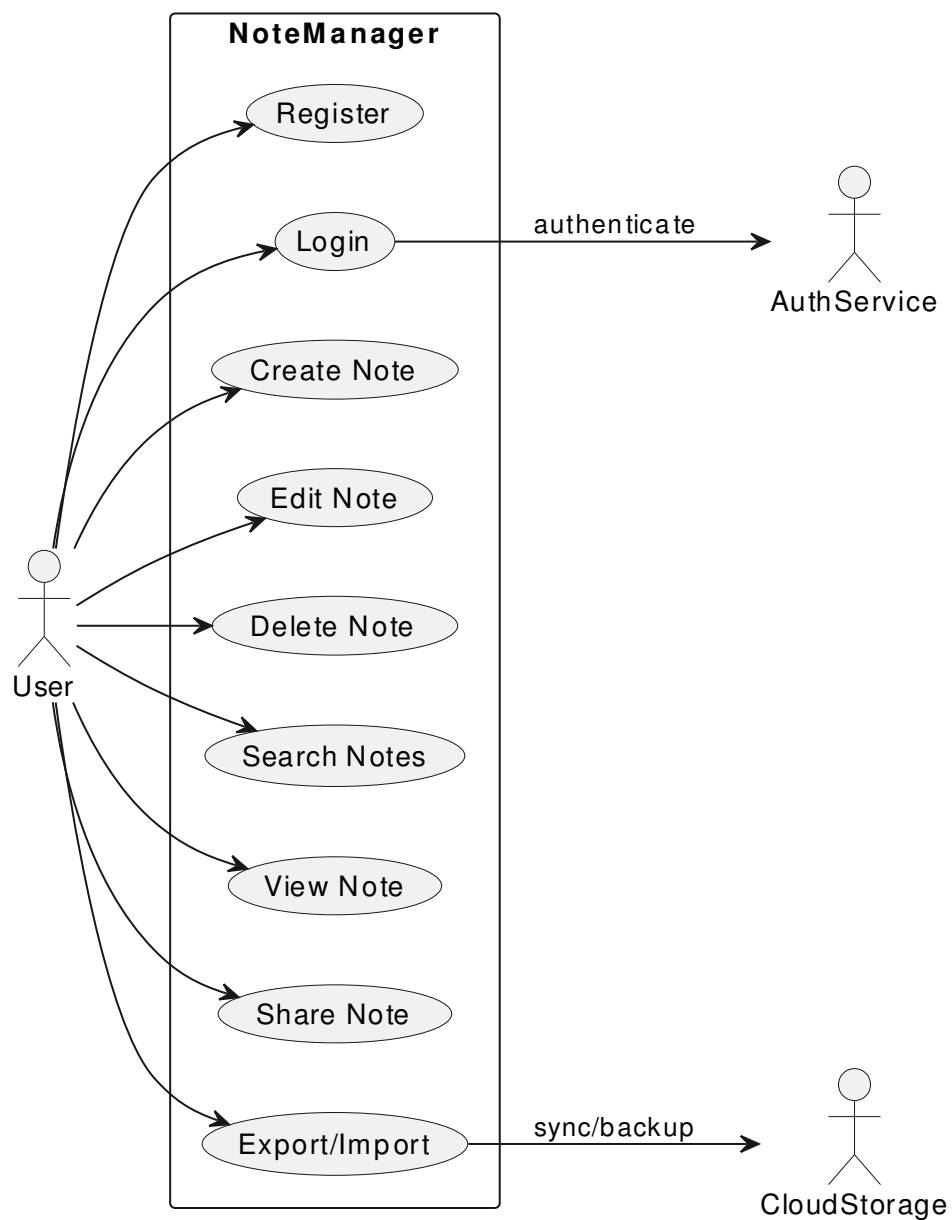
Ход работы.

Акторы:

- User — обычный пользователь заметок.
- AuthService — внешний провайдер аутентификации (OIDC/пароль).
- CloudStorage — внешнее хранилище для синхронизации (опционально).

Ключевые варианты использования:

- Зарегистрироваться / Войти
- Создать заметку
- Редактировать заметку
- Удалить заметку
- Найти заметки
- Просмотреть заметку
- Поделиться заметкой (ссылка/права)
- Экспорт/Импорт (Markdown/JSON)

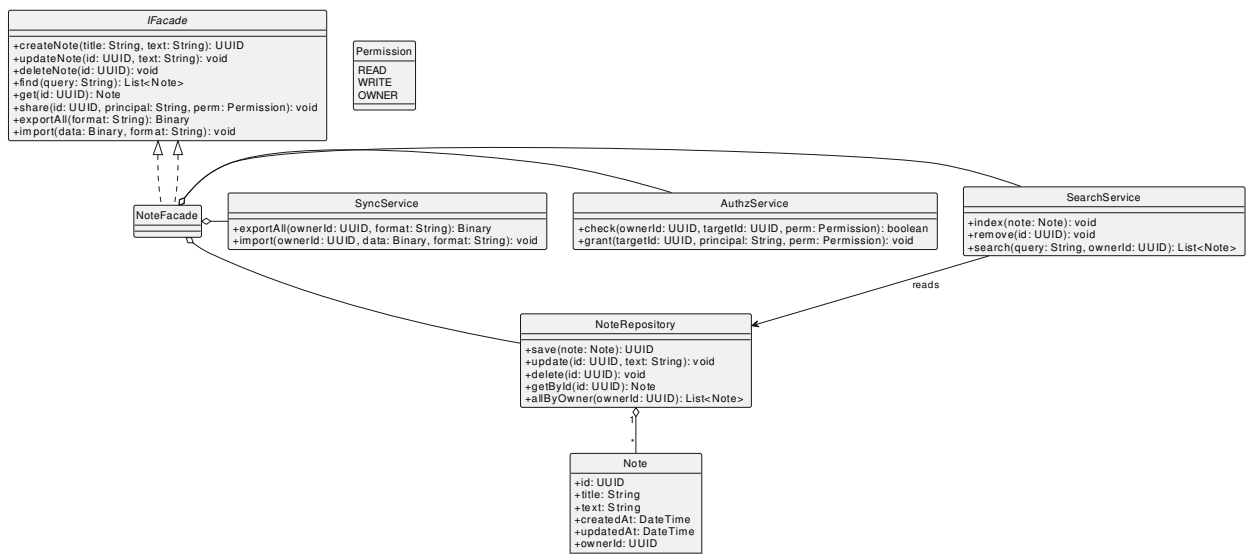


2. Диаграмма классов (Class Diagram)

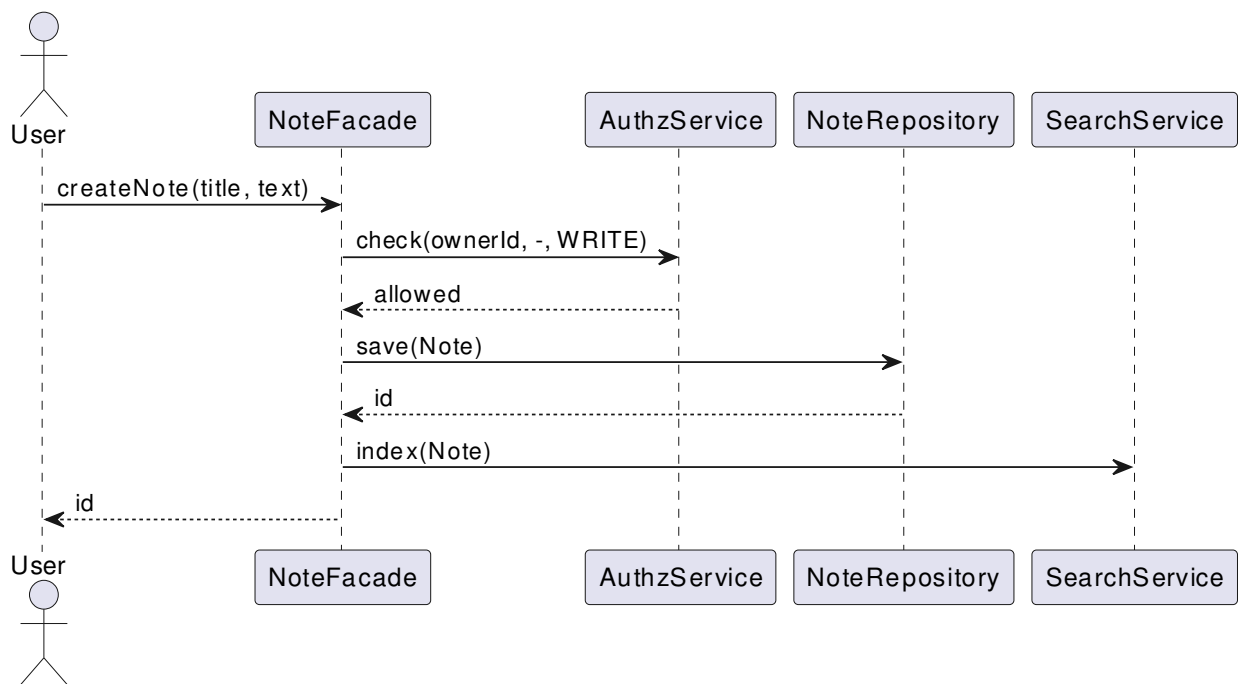
Замысел: фасад предоставляет единый, стабильный API для UI/клиентов; за фасадом — репозиторий заметок, поиск, сервис прав/доступа, синхронизация.

Ключевые классы:

- IFacade / NoteFacade — внешний интерфейс операций.
- Note — доменная сущность.
- NoteRepository — CRUD и выборки.
- SearchService — индексирование/поиск по тексту.
- AuthzService — проверка доступа/ролей.
- SyncService — экспорт/импорт/синхронизация (облачное хранилище).



Краткая декомпозиция сценария (последовательность createNote).



Выбор методологии управления проектом.

Для «NoteManager» целесообразен Scrum: есть динамически меняющиеся пользовательские ожидания (поиск, шэринг, офлайн, синхронизация), инкрементальная ценность видна каждые 1–2 недели, легко поддерживать бэклог фич и приоритизацию.

Почему не Kanban: хорош для потока эксплуатации/поддержки; у нас — продуктовая разработка с итеративными инкрементами.

Почему не чистый XP: инженерные практики XP (TDD, CI/CD, парное программирование) берём внутрь Scrum-процесса.

Вывод.

Построены диаграмма вариантов использования и диаграмма классов системы «NoteManager». Определены акторы, ключевые сценарии и сервисная архитектура с фасадом. Выбрана методология Scrum для итеративной реализации и управляемого расширения функциональности.

Приложение.

Use case.

@startuml

left to right direction

actor User

actor AuthService

actor CloudStorage

rectangle NoteManager {

 usecase "Register" as UC_Register

 usecase "Login" as UC_Login

 usecase "Create Note" as UC_Create

 usecase "Edit Note" as UC_Edit

 usecase "Delete Note" as UC_Delete

 usecase "Search Notes" as UC_Search

 usecase "View Note" as UC_View

 usecase "Share Note" as UC_Share

 usecase "Export/Import" as UC_Export

User --> UC_Register

User --> UC_Login

User --> UC_Create

User --> UC_Edit

User --> UC_Delete

User --> UC_Search

User --> UC_View

User --> UC_Share

User --> UC_Export

```

UC_Login --> AuthService : authenticate
UC_Export --> CloudStorage : sync/backup
}
@enduml

```

Class Diagram.

```

@startuml

```

```

hide circle

```

```

skinparam classAttributeIconSize 0

```

```

interface IFacade {
    +createNote(title: String, text: String): UUID
    +updateNote(id: UUID, text: String): void
    +deleteNote(id: UUID): void
    +find(query: String): List<Note>
    +get(id: UUID): Note
    +share(id: UUID, principal: String, perm: Permission): void
    +exportAll(format: String): Binary
    +import(data: Binary, format: String): void
}

```

```

class NoteFacade implements IFacade

```

```

class Note {
    +id: UUID
    +title: String
    +text: String
    +createdAt: DateTime
    +updatedAt: DateTime
    +ownerId: UUID
}

```

```
}
```

```
enum Permission { READ; WRITE; OWNER }
```

```
class NoteRepository {  
    +save(note: Note): UUID  
    +update(id: UUID, text: String): void  
    +delete(id: UUID): void  
    +getById(id: UUID): Note  
    +allByOwner(ownerId: UUID): List<Note>  
}
```

```
class SearchService {  
    +index(note: Note): void  
    +remove(id: UUID): void  
    +search(query: String, ownerId: UUID): List<Note>  
}
```

```
class AuthzService {  
    +check(ownerId: UUID, targetId: UUID, perm: Permission): boolean  
    +grant(targetId: UUID, principal: String, perm: Permission): void  
}
```

```
class SyncService {  
    +exportAll(ownerId: UUID, format: String): Binary  
    +import(ownerId: UUID, data: Binary, format: String): void  
}
```

```
IFacade <|.. NoteFacade
```

```
NoteFacade o- NoteRepository
```

NoteFacade o- SearchService
NoteFacade o- AuthzService
NoteFacade o- SyncService
SearchService --> NoteRepository : reads
NoteRepository "1" o-- "*" Note
@enduml

Последовательность createNote.

@startuml
actor User
participant NoteFacade
participant AuthzService
participant NoteRepository
participant SearchService

User -> NoteFacade: createNote(title, text)
NoteFacade -> AuthzService: check(ownerId, -, WRITE)
AuthzService --> NoteFacade: allowed
NoteFacade -> NoteRepository: save(Note)
NoteRepository --> NoteFacade: id
NoteFacade -> SearchService: index(Note)
NoteFacade --> User: id
@enduml