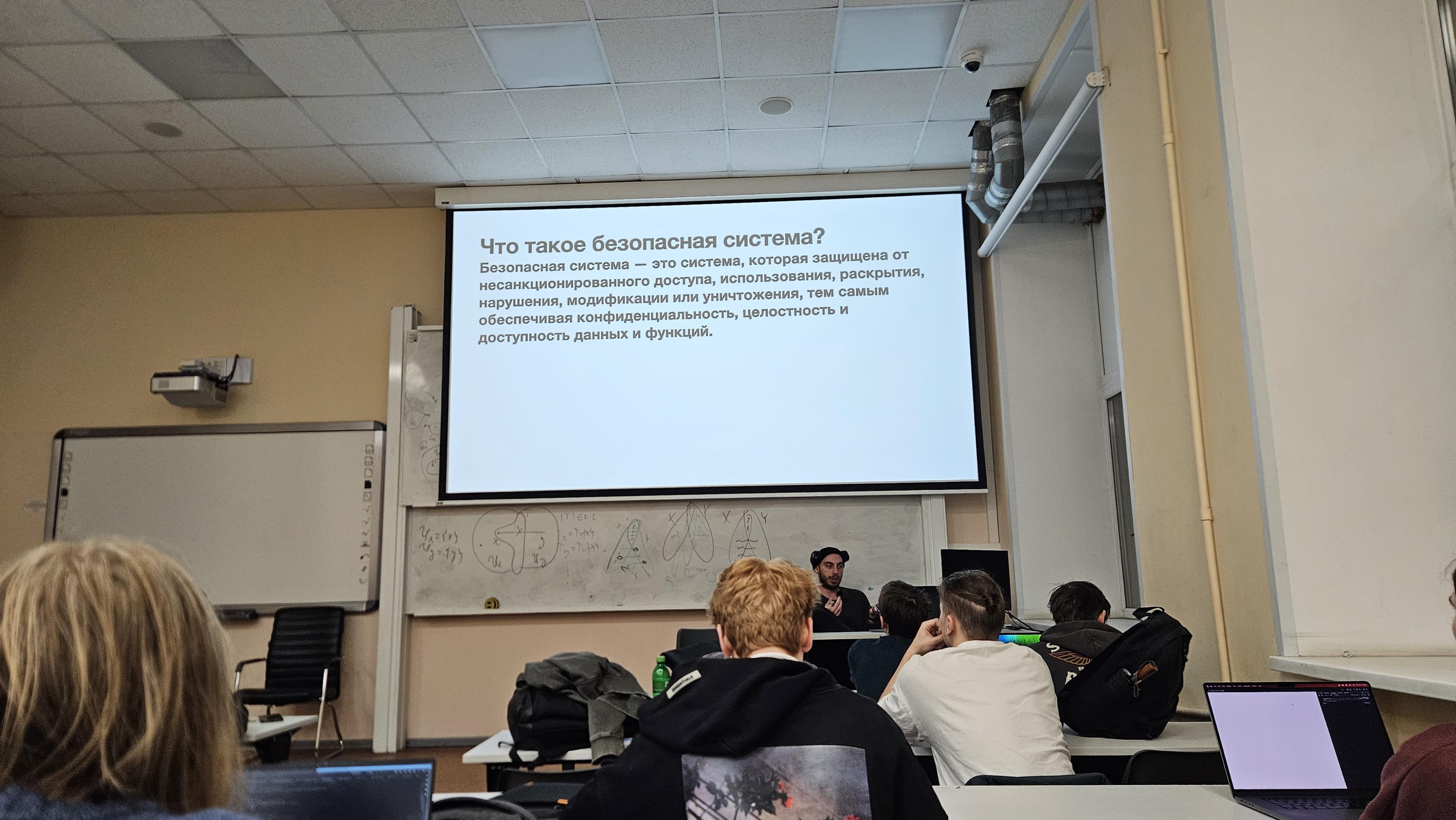
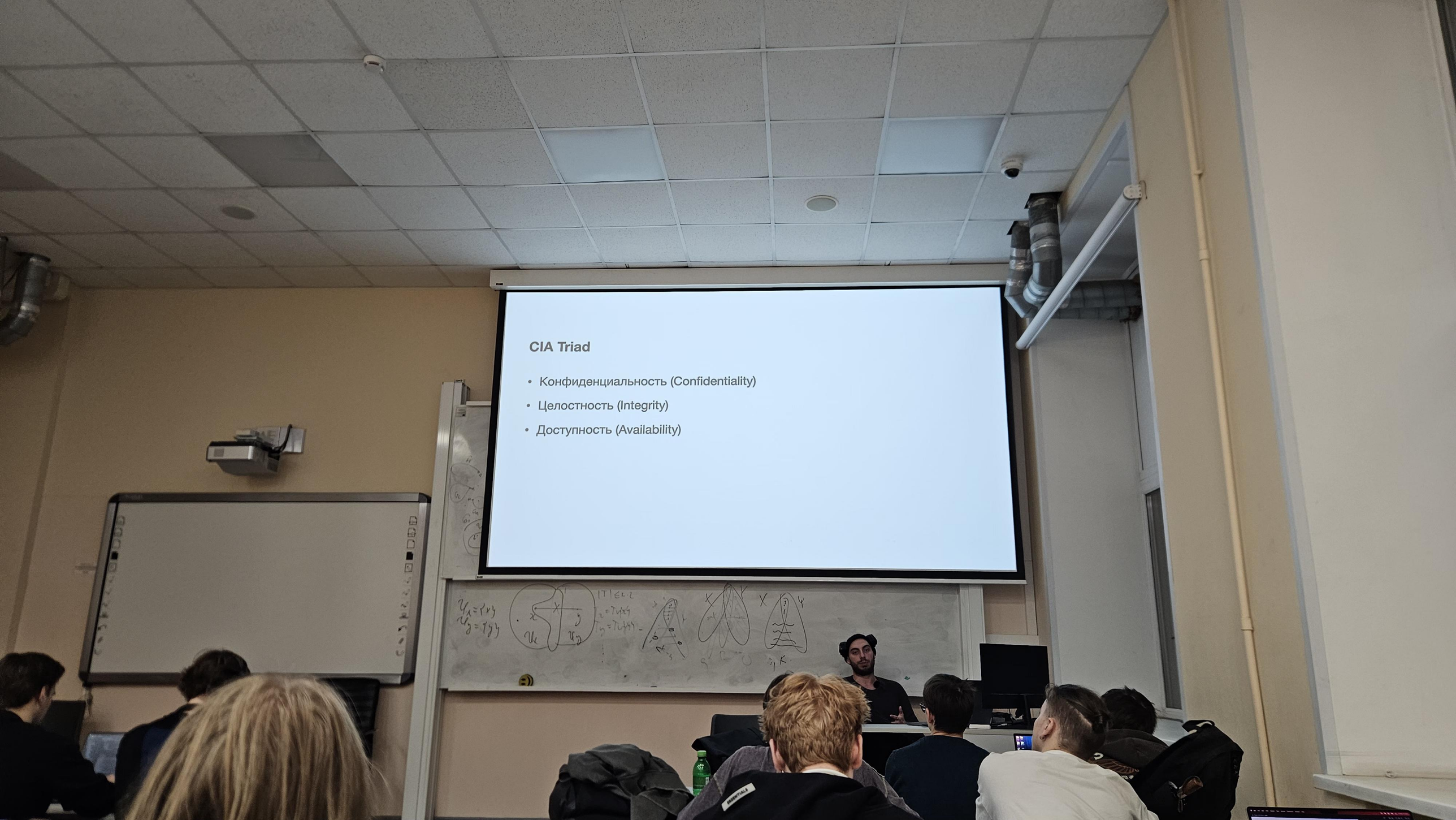
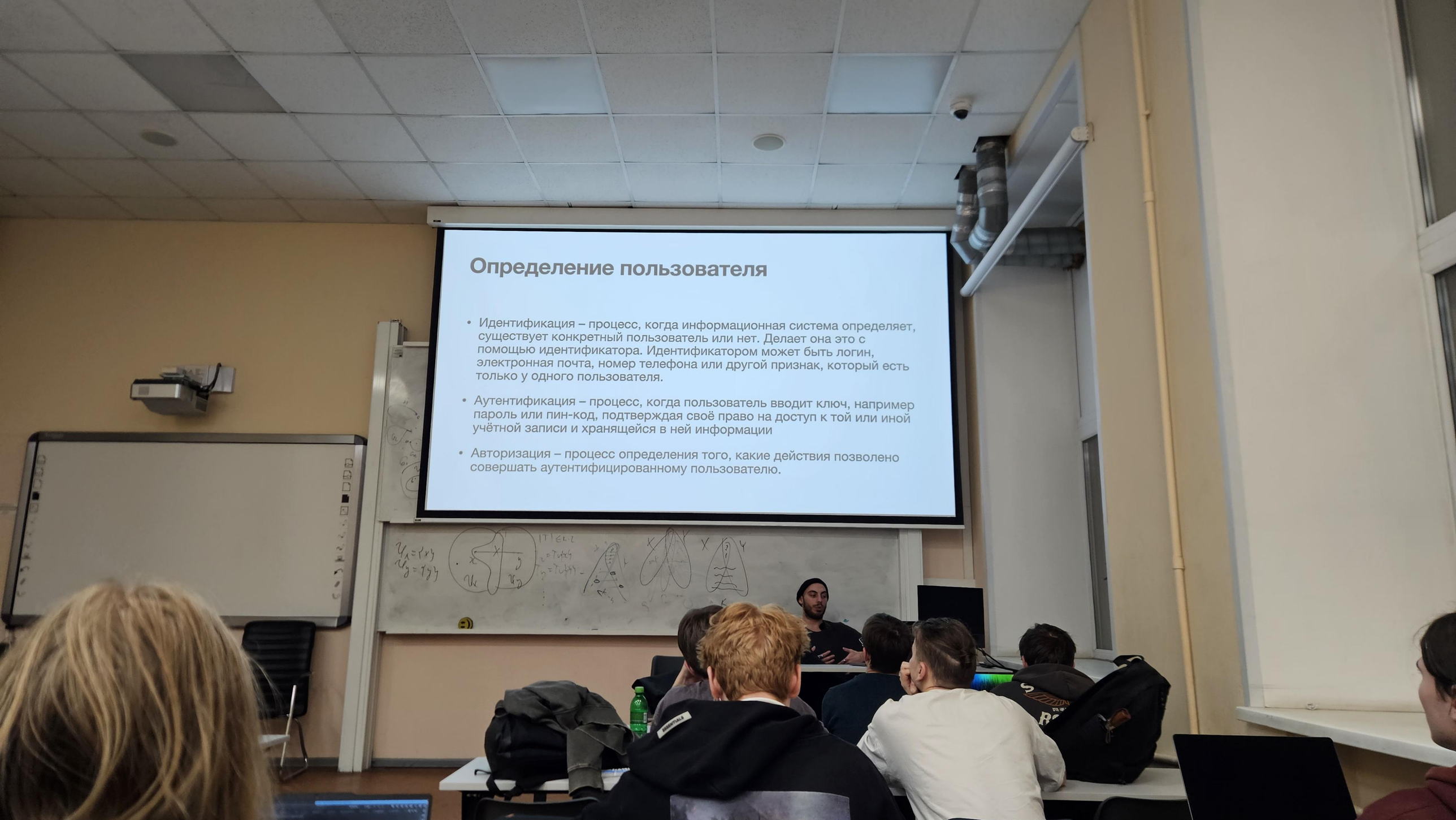
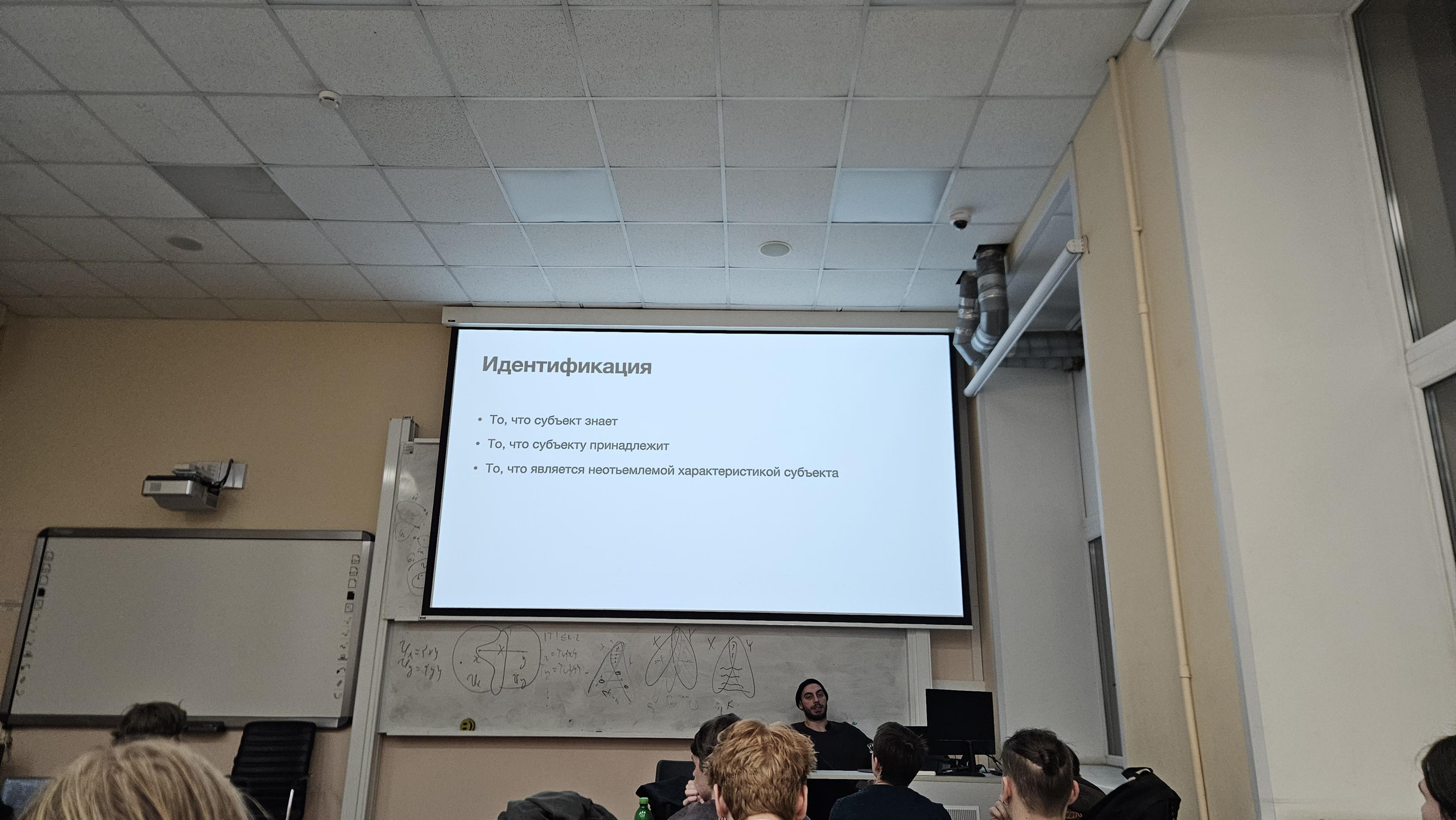

Что такое безопасная система?
Безопасная система — это система, которая защищена от несанкционированного доступа, использования, раскрытия, нарушения, модификации или уничтожения, тем самым обеспечивая конфиденциальность, целостность и доступность данных и функций.


CIA Triad
• Конфиденциальность (Confidentiality)
• Целостность (Integrity)
• Доступность (Availability)


Определение пользователя
Идентификация – процесс, когда информационная система определяет, существует конкретный пользователь или нет. Делает она это с помощью идентификатора. Идентификатором может быть логин, электронная почта, номер телефона или другой признак, который есть только у одного пользователя.
Аутентификация – процесс, когда пользователь вводит ключ, например пароль или пин-код, подтверждая своё право на доступ к той или иной учётной записи и хранящейся в ней информации
Авторизация – процесс определения того, какие действия позволено совершать аутентифицированному пользователю.


Идентификация
То, что субъект знает
То, что субъекту принадлежит
То, что является неотьемлемой характеристикой субъекта

Многофакторная идентификация

- Если для входа требуются пароль и, например, ответ на секретный вопрос (который тоже является "знанием"), то это не многофакторная аутентификация, а всего лишь двухэтапная проверка в рамках одного фактора.

Авторизация

	User 1	User 2	User 3
Object 1	READ		
Object 2		READ, ADD	
Object 3		UPDATE, READ	
Object 4	READ		

Дискретный уровень доступа

- Каждый объект имеет владельца
- Владелец полностью контролирует доступ к своему объекту
- Права доступа определяются на основе списков контроля доступа (ACL)

- SELECT *
- FROM pg_class

```
[null]
[null]
robin155=arwdDxt/robin155
robin155=arwdDxt/robin155,=r//robin155
[null]
[null]
```

- SYSTEM PRIVILEGES
- DATABASE PRIVILEGES
- SCHEMA PRIVILEGES
- TABLE PRIVILEGES
- COLUMN PRIVILEGES
- ROW-LEVEL PRIVILEGES

Модель доступа

Пользователи и группы

- Реализована через концепцию ролей, которая служит для управления доступом и правами пользователей к объектам базы данных. В этой системе каждая роль может выступать как в роли пользователя, так и в роли группы пользователей.

Модель доступа

Роли

- В PostgreSQL понятие “пользователь” является частным случаем роли. Каждая роль может иметь или не иметь возможность входа в систему (login privilege). Если у роли установлено свойство LOGIN, она может использоваться как учетная запись пользователя.
- Роли могут быть членами других ролей. Это позволяет организовать группы пользователей — например, создать роль developers, которой будут принадлежать все разработчики, и назначить этой роли общие привилегии.

Модель доступа

Роли

- Создание CREATE ROLE {Name} {attr1, attr2, …}
- Изменение ALTER ROLE developer WITH PASSWORD 'newsecret';
- LOGIN
- SUPERUSER
- CREATEDB
- CREATEROLE
- PASSWORD 'secret'

Модель доступа

Наследование привилегий

- роль, являющаяся членом другой роли, по умолчанию автоматически наследует все привилегии (права доступа) этой родительской роли.
- GRANT role1 to role2
- REVOKE role1 FROM role2

Модель доступа

Смена владельца

- ALTER {obj} OWNER TO {role1}
- Владеет роли и роли включающиеся в нее

Модель доступа

Привилегии таблиц

- SELECT
- INSERT
- UPDATE
- REFERENCES
- DELETE
- TRUNCATE
- TRIGGER

Оффтоп

Truncate

- TRUNCATE не проверяет каждую строку для удаления, а просто освобождает данные, удаляя содержимое таблицы целиком.
- Вместо того, чтобы логировать каждое удаление строки, TRUNCATE записывает в журнал меньший объем информации.
- При использовании TRUNCATE не вызываются триггеры, определенные для операций удаления.
- Если TRUNCATE выполняется в транзакционном блоке, то операция может быть отменена командой ROLLBACK до фиксации транзакции.

Модель доступа

Привилегии БД

- CREATE
- CONNECT
- TEMPERARY

Модель доступа

Привилегии схем

- CREATE
- USAGE
- Права доступа к схеме влияют на возможность создания новых объектов, но сами по себе не контролируют доступ к уже существующим объектам. Для этого необходимо назначать соответствующие привилегии непосредственно на уровне объектов.

Оффтоп

Схема public

- Создаётся по умолчанию при создании новой базы данных.
- Объекты, в ней, доступны всем пользователям, если не настроены отдельные ограничения.
- При создании объектов не указана какая-либо другая схема, они автоматически создаются в public.

Оффтоп

Стандартные БД

- Когда создаётся новая база данных без явного указания шаблона, используется база данных template1.
- Template0 — это «чистая» база данных, которая не изменялась после установки PostgreSQL. Она содержит минимальное количество объектов и служит эталоном.

Модель доступа

Привилегии БД

- CREATE
- CONNECT
- TEMPERARY

Модель доступа

Привилегии схем

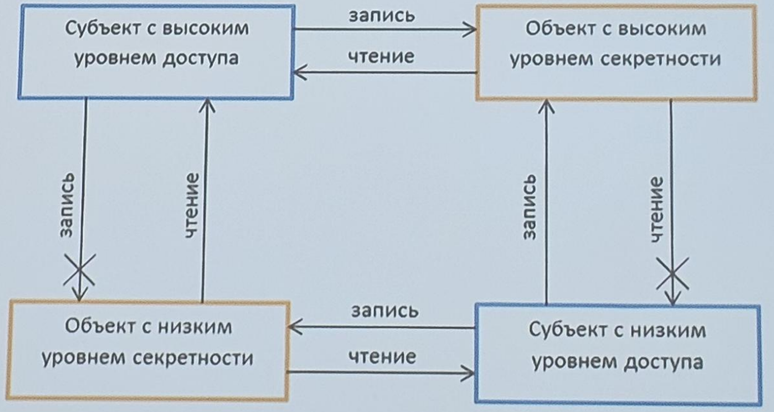
- CREATE
- USAGE
- Права доступа к схеме влияют на возможность создания новых объектов, но сами по себе не контролируют доступ к уже существующим объектам. Для этого необходимо назначать соответствующие привилегии непосредственно на уровне объектов.

Оффтоп

Схема public

- Создаётся по умолчанию при создании новой базы данных.
- Объекты, в ней, доступны всем пользователям, если не настроены отдельные ограничения.
- При создании объектов не указана какая-либо другая схема, они автоматически создаются в public.

Мандатный доступ



- "Нет права давать права"
- Протоколирование ВСЕХ действий
- Защита от копирования

Шифрование БД

- Прозрачное шифрование
- На уровне столбцов
- На уровне приложения

УГРОЗА ДЕЙСТВИЙ ПРИВИЛЕГИРОВАННЫХ ПОЛЬЗОВАТЕЛЕЙ

- Принцип разделения обязанностей
- Принцип наименьших привилегий

Плюсы

- Инкапсулирование функциональности
- Изоляция пользователей от таблиц.
- В некоторых случаях ускорение выполнения (Предкомпиляция)

В чем отличие?

- Функция имеет возвращаемый тип и возвращает значение
- Использование DML запросов внутри функции невозможно. В функциях разрешены только SELECT-запросы
- Вызов хранимой процедуры из функции невозможно
- Вызов функции внутри SELECT запросов возможен

- Может не возвращать значение или возвращать через OUTPUT параметры
- Использование DML-запросов возможно в хранимой процедуре.
- Использование или же управление транзакциями возможно в хранимой процедуре
- Вызов хранимой процедуры из SELECT запросов невозможно

Создание функции

```sql
CREATE OR REPLACE FUNCTION get_table_count()
RETURNS BIGINT
AS $$
    SELECT COUNT(*) FROM your_table_name;
$$ LANGUAGE SQL;
```

Создание функции

```sql
CREATE OR REPLACE FUNCTION get_student_id(_group_id integer)
RETURNS TABLE(student_id integer)
AS $$
BEGIN
    RETURN QUERY
    SELECT id FROM student
    WHERE group_id = _group_id;
END;
$$ LANGUAGE plpgsql;

-- Вызов функции
SELECT * FROM get_student_id(2);
```

Функции

- Функции могут состоять из нескольких операторов SQL.
- Возвращается значение из RETURN
- Нельзя использовать операторы управления транзакции (BEGIN, COMMIT, ROLLBACK)
- Нельзя использовать служебные команды (CREATE INDEX)

Создание функции

```
CREATE FUNCTION add(int, int, int) RETURNS int AS $$
BEGIN
    RETURN $1 + $2 + $3;
END;
$$ LANGUAGE plpgsql;
```

Типы переменных

- IN
- OUT
- INOUT
- RETURN

Привелегии доступа

- Definer — Создателя
- Invoker — Вызывающего

Перегрузка операторов

```sql
CREATE FUNCTION add(IN a int, IN b int, IN c int DEFAULT 0) RETURNS int AS $$
    SELECT a + b + c;
$$ LANGUAGE sql;

CREATE FUNCTION add(IN a float, IN b int) RETURNS float AS $$
    SELECT a + b;
$$ LANGUAGE sql;

CREATE FUNCTION add(IN a float, IN b float) RETURNS float AS $$
    SELECT a + b;
$$ LANGUAGE sql;
```

Полиморфизм

```
CREATE FUNCTION add(IN a anyelement, IN b anyelement, IN c anyelement) RETURNS anyelement
as $$
SELECT a + b + c;
$$ LANGUAGE sql;
```

Категории изменчивости

- Volatile — возвращаемое значение может быть произвольно менять на одних и тех же входных.
- Stable — Значение не меняется в рамках одного оператора, функция не может менять таблицы (Оптимизатор может **кэшировать** результаты в пределах команды)
- Immutable — значение не меняется, функция детерминирована, функция не может менять таблицы (Можно вызывать на этапе **планирования запроса)**

Создание процедуры с возвращаемым значением

```
CREATE PROCEDURE pay_in() RETURN int
AS  $$
    UPDATE student SET grant = grant + 200;
    SELECT max(grant) FROM student
$$ LANGUAGE sql;

CALL pay_in();
```

PL/pgSQL

это загружаемый процедурный язык для системы управления базами данных.

- может быть использован для создания функций, процедур и триггеров
- добавляет структуры управления к языку SQL
- наследует все пользовательские типы, функции, процедуры и операторы
- прост в использовании

Форма блока

```
-- Метка
DECLARE
-- Описание переменных
BEGIN
-- Тело
EXCEPTION
-- Обработка ошибок
END
```

STRICT

```
BEGIN
    SELECT * INTO STRICT var FROM student WHERE student_name = my_name;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE EXCEPTION 'Студент % не найден', my_name;
        WHEN TOO_MANY_ROWS THEN
            RAISE EXCEPTION 'Студентов больше одного';
END;
```

DIAGNOSTICS

```
DECLARE
    r record;
    count int;
BEGIN
    SELECT id, code INTO r FROM table;

    GET DIAGNOSTICS count = row_count;
    RAISE NOTICE 'rowcount = %', count;
END
```

```
<<outer_block>>
DECLARE
    a := 10;
BEGIN
    <<inner_block>>
    DECLARE
        a := 2
        RAISE NOTICE '%', outer_block.a + inner_blcok.a;
    END inner_block;
END outer_block
```

Подпрограммы PL/pgSQL

```
CREATE FUNCTION add(IN a int, IN b int) RETURNS int
AS $$
BEGIN
    RETURN a + b;
END;
$$ LANGUAGE plsql IMMUTABLE;
```

Триггеры

Используется, когда мы хотим действия в момент изменения данных.

```
CREATE [ OR REPLACE ] [ CONSTRAINT ] TRIGGER имя { BEFORE | AFTER | INSTEAD OF } { событие [ OR ... ]
}
    ON имя_таблицы
    [ FROM ссылающаяся_таблица ]
    [ NOT DEFERRABLE | [ DEFERRABLE ] [ INITIALLY IMMEDIATE | INITIALLY DEFERRED ] ]
    [ REFERENCING { { OLD | NEW } TABLE [ AS ] имя_переходного_отношения } [ ... ] ]
    [ FOR [ EACH ] { ROW | STATEMENT } ]
    [ WHEN ( условие ) ]
    EXECUTE { FUNCTION | PROCEDURE } имя_функции ( аргументы )
```

Триггер

- BEFORE/AFTER - Выполняется перед дейтствием или после. Если используем перед, то мы можем менять данные.
- FROM - на какую то таблицу
- FOR ROW/STATEMENT - На весь запрос целиком или на каждую строчку. У стейтмент триггера нет информации об измененных данных
- WHEN - содержит условие, определяющее будет ли вызываться функция или нет.

Переменные триггера

- NEW - Содержит новую строку
- OLD - Содержит старую строку
- TG_NAME - Содержит название триггера
- TG_OP - Содержит название операции
- TG_TABLE_NAME - Содержит название таблицы

Пример триггера

```
CREATE TRIGGER check_update
    BEFORE UPDATE ON accounts
    FOR EACH ROW
    WHEN (OLD.balance IS DISTINCT FROM NEW.balance)
    EXECUTE FUNCTION check_account_update();
```

ConsoleApp6 > ConsoleApp6 > Week.cs > Week > GetEnumerator

```
public class Week : IEnumerable<string>
{
    private readonly string[] _days = [
        "Понедельник",
        "Вторник",
        "Среда",
        "Четверг",
        "Пятница",
        "Суббота",
        "Воскресенье"
    ];

    1 usage
    public IEnumerator<string> GetEnumerator()
    {
        return new WeekEnumerator
    }

    IEnumerator IEnumerable.GetEnumerator()
    {
```

19:34 CRLF UTF-8 2 errors in 1 file 4 spaces

ConsoleApp6 · Version control

Solution · Files

ConsoleApp6 · 1 projec
ConsoleApp6
Dependencies
Program.cs
Week.cs
Scratches and Console

Program.cs · Week.cs · IEnumerator.cs · IDisposable.cs

```
public class WeekEnumerator : IEnumerator<string>
    public bool MoveNext()
        _index++;
        return _index < _days.Length;
    }

    public void Reset() => throw new NotSupportedException();

    string IEnumerator<string>.Current
    {
        get
        {
            if (_index == -1)
                throw new InvalidOperationException("");
            return _days[_index];
        }
    }

    object? IEnumerator.Current => ;

    public void Dispose()
```

ConsoleApp6 > ConsoleApp6 > Week.cs > WeekEnumerator > Current

6 mins · 51:57 · CRLF · UTF-8 · 2 errors in 1 file · 4 spaces

ENG · 11:50 · 15.11.2025

ConsoleApp6 · Version control

Solution Files

- ConsoleApp6 · 1 projec
  - ConsoleApp6
    - Dependencies
    - Program.cs
    - Week.cs
    - WeekEnumerato
  - Scratches and Consol

Program.cs | Week.cs | WeekEnumerator.cs | IEnumerator.cs | IDisposable.cs

```
5   public class Week : IEnumerable<string>
8       {
9           "Понедельник",
10          "Вторник",
11          "Среда",
12          "Четверг",
13          "Пятница",
14          "Суббота",
15      ];  "Воскресенье"
16      ];
17      1 usage
18      public IEnumerator<string> GetEnumerator()
19      {   return new WeekEnumerator(_days);
20      }
21      IEnumerator IEnumerable.GetEnumerator()
22      {   return GetEnumerator();
23      }
24  }
25  }
```

ConsoleApp6 > ConsoleApp6 > Week.cs

6 mins 26:2 CRLF UTF-8 4 spaces

ConsoleApp6 · Version control

Solution · Files

- ConsoleApp6 · 1 projec
  - ConsoleApp6
    - Dependencies
    - Program.cs
    - Week.cs
    - WeekEnumerato
  - Scratches and Consol

Program.cs · Week.cs · WeekEnumerator.cs · IEnumerator.cs · IDisposable.cs

```
public class Week : IEnumerable<string>
        "Четверг",
        "Пятница",
        "Суббота",
        "Воскресенье"
    ];

    1 usage
    public IEnumerator<string> GetEnumerator()
    {
        return new WeekEnumerator(_days);
    }

    IEnumerator IEnumerable.GetEnumerator()
    {
        return GetEnumerator();
    }
}
```

ConsoleApp6 > ConsoleApp6 > Week.cs > Week

6 mins · 25:6 · CRLF · UTF-8 · 4 spaces

ConsoleApp6 · Version control

Solution · Files

- ConsoleApp6 · 1 projec
  - ConsoleApp6
    - Dependencies
    - Program.cs
    - Week.cs
    - WeekEnumerato
  - Scratches and Consol

Program.cs · Week.cs · WeekEnumerator.cs · IEnumerator.cs · IDisposable.cs

```
using System.Collections;

namespace ConsoleApp6;

public class Week : IEnumerable<string>
{
    private readonly string[] _days = [
        "Понедельник",
        "Вторник",
        "Среда",
        "Четверг",
        "Пятница",
        "Суббота",
        "Воскресенье"
    ];

    1 usage
    public IEnumerator<string> GetEnumerator()
    {
        return new WeekEnumerator(_days);
    }
```

ConsoleApp6 > ConsoleApp6 > Week.cs > Week

6 mins · 25:6 · CRLF · UTF-8 · 4 spaces

ConsoleApp6 · Version control

Solution · Files

- ConsoleApp6 · 1 projec
  - ConsoleApp6
    - Dependencies
    - Program.cs
    - Week.cs
    - WeekEnumerato
  - Scratches and Consol

Program.cs · Week.cs · WeekEnumerator.cs · IEnumerator.cs · IDisposable.cs

```
{}
// See https://aka.ms/new-console-template for more information

internal class Program
{
    public static void Main(string[] args)
    {
        List<int> numbers = [1, 2, 3, 4, 5];
        foreach (var n:int in numbers)
        {
            Console.WriteLine(n);
        }
    }
}
```

ConsoleApp6 > ConsoleApp6 > Program.cs > Program > Main

6 mins · 10:14 · CRLF · UTF-8 · 4 spaces

ConsoleApp6 · Version control

Solution · Files

- ConsoleApp6 · 1 proje
  - ConsoleApp6
    - Dependencies
    - Program.cs
    - Week.cs
    - WeekEnumerato
  - Scratches and Consol

Program.cs · Week.cs · WeekEnumerator.cs · IEnumerator.cs · IDisposable.cs

```
// See https://aka.ms/new-console-template for more information

using ConsoleApp6;

internal class Program
{
    public static void Main(string[] args)
    {
        foreach (var n :string  in new Week())
        {
            Console.WriteLine(n);
        }
    }
}
```

ConsoleApp6 > ConsoleApp6 > Program.cs > Program

6 mins · 13:6 · CRLF · UTF-8 · 4 spaces

ENG · 11:54 · 15.11.2025

ConsoleApp6 · Version control

Solution · Files

- ConsoleApp6 · 1 project
  - ConsoleApp6
    - Dependencies
    - Strategies
      - DayOfWeekSelect
      - DayOfWeekSelect
      - FromMondayDayC
      - IDayOfWeekSelec
    - Program.cs
    - Week.cs
    - WeekEnumerator.cs
- Scratches and Consoles

Week.cs · WeekEnumerator.cs · IDayOfWeekSelector.cs · FromMondayDayOfWeekSelector.cs · DayOfWeek

```
6   public class WeekEnumerator : IEnumerator<string>
    1 usage
27      private string GetDay()
28      {
29          return _index == -1
30              ? throw new InvalidOperationException("You should call MoveNext() first.")
31              : _days[_index];
32      }
33
34      string IEnumerator<string>.Current => GetDay();
35      object? IEnumerator.Current => (this as IEnumerator<string>).Current;
36
37      public void Dispose()
38      {
39      }
40  }
```

ConsoleApp6 > ConsoleApp6 > WeekEnumerator.cs > WeekEnumerator > GetDay

6 mins · 31:29 · CRLF · UTF-8 · 4 spaces

ConsoleApp6 · Version control

Solution Files

- C:\temp\ConsoleApp6
  - .idea
  - ConsoleApp6
    - bin
      - Debug
        - net9.0
          - ConsoleApp6.dll
          - ConsoleApp6.exe
          - ConsoleApp6.pdb
          - ConsoleApp6.deps.json
          - ConsoleApp6.runtimeconfig.json
          - snapshot.txt
    - obj
    - Observables
    - Strategies
    - ConsoleApp6.csproj
    - DuckEnumerable.cs
    - Program.cs
    - Week.cs
    - WeekEnumerator.cs
  - ConsoleApp6.sln · Active
  - ConsoleApp6.sln.DotSettings.user
- Scratches and Consoles

Program.cs | AlwaysFaultingObserver.cs | FlushToConsoleOnUpdateObserver.cs | IMyObs

```
using ConsoleApp6.Strategies;

internal class Program
{
    public static void Main(string[] args)
    {
        var week = new Week();
        week.AddObserver(new AlwaysFaultingObserver());
        week.AddObserver(new FlushToConsoleOnUpdateObserver<object>());
        week.SnapshotManager.LoadFromStorage();
        foreach (var n :string  in week)
        {
            Console.Write($"{n}, ");
        }
        week.SetSelector(DayOfWeekSelectors.FromSunday);
        week.SnapshotManager.PersistSnapshot();
        Console.WriteLine();
        foreach (var n :string  in week)
        {
            Console.Write($"{n}, ");
        }
    }
```

ConsoleApp6 > ConsoleApp6 > Program.cs > Program > Main

59 mins 16:10 CRLF UTF-8 4 spaces

ENG 13:03 15.11.2025