Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

**ОТЧЕТ ПО ИТОГОВОМУ ПРОЕКТУ**

по дисциплине «Базы данных»

**Выполнил:**

студент группы 5130902/20201 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К. Р. Айтынбетова

подпись

**Проверил:**

кандидат тех. наук, доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. А. Нестеров

подпись

«\_\_\_»\_\_\_\_\_\_\_ 2025г.

Санкт-Петербург, 2025

Этап 1

Делаем описание предметной области, выделяем правила и ограничения. Описываем, что должно делать приложение, иллюстрируем это UML-диаграммами вариантов использования (use case).

Описание предметной области

Система предназначена для управления художественной галереей. Пользователи могут просматривать текущие выставки, искать произведения искусства по критериям (художник, стиль, год, название) и сохранять понравившиеся работы в избранное. Администраторы управляют выставками: добавляют, редактируют, удаляют информацию о них.

Правила и ограничения

1. Каждая выставка может содержать несколько произведений искусства, произведение может участвовать в нескольких выставках (связь многие-ко-многим).
2. У каждого произведения есть уникальный идентификатор, художник, стиль и год создания.
3. Пользователи могут добавлять произведения в избранное, для этого требуется учетная запись.
4. Администраторы имеют отдельные учетные данные для доступа к функциям управления.
5. Поиск производится по неполным данным (например, части имени художника).

Use-case:



Рисунок 1 – Use-case диаграмма

Этап 2

Строим логическую модель базы данных, определяем атрибуты, идентификаторы, связи. Если понадобится - супертипы или/и арки.

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 – Логическая модель

Этап 3

Переходим к реляционной модели. Уточняем типы данных (они должны быть максимально соответствующими хранимым данным: не надо числовые идентификаторы хранить в строковых типах или фамилию хранить как большой текстовый объект), разрешаем связи многие-ко-многим, описываем ограничения (используем Data Modeler, ERwin, DBeaver, Workbench, Ред Эксперт).

Доказываем, что все таблицы соответствуют НФБК.

Реляционная модель:

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3 – Реляционная модель

Отношение находится в нормальной форме БойсаКодда (НФБК), тогда и только тогда, когда все детерминанты нетривиальных и неприводимых функциональных зависимостей являются потенциальными ключами.

Докажем, что БД соответствует НФБК:

1. Таблица artist

* Первичный ключ: id.
* Функциональные зависимости:
  + id → name, country, birth\_year, death\_year.  
    Все атрибуты зависят только от суперключа (id).

2. Таблица style

* Первичный ключ: id.
* Функциональные зависимости:
  + id → name,
  + name → id (поскольку name уникален).  
    Обе зависимости основаны на суперключах (id и name).

3. Таблица artwork

* Первичный ключ: id.
* Функциональные зависимости:
  + id → title, year, image\_path, artist\_id, style\_id.
  + artist\_id → artist(id) (внешний ключ).
  + style\_id → style(id) (внешний ключ).

Все атрибуты зависят только от суперключа (id). Внешние ключи не создают новых функциональных зависимостей внутри таблицы.

4. Таблица exhibition

* Первичный ключ: id.
* Функциональные зависимости:
  + id → name, description.  
    Нет зависимостей от неключевых атрибутов.

5. Таблица exhibition\_artwork

* Первичный ключ: (exhibition\_id, artwork\_id).

Таблица содержит только ключи, функциональные зависимости отсутствуют.

6. Таблица user

* Первичный ключ: id.
* Функциональные зависимости:
  + id → username, email, password,
  + username → id (так как username уникален),
  + email → id (так как email уникален).

Все зависимости основаны на суперключах (id, username, email).

7. Таблица favorite

* Первичный ключ: (user\_id, artwork\_id).

Нет неключевых атрибутов.

8. Таблица admin

* Первичный ключ: id.
* Функциональные зависимости:
  + id → login, password
  + login → id (так как login уникален).

Все зависимости основаны на суперключах.

Этап 4

Создаем базу данных в среде одной из реляционных СУБД: PostgreSQL (в этом случае рекомендуется использовать БД в облаке Postgres Pro, чтобы было проще разрабатывать дома и показывать в классе), Oracle Database, Microsoft SQL Server, MySQL (если будут желающие, наверное можно будет на два-три месяца получить сервер MySQL в Yandex.Cloud), Ред БД / FireBird.

При необходимости, добавляем ограничения, не учтенные в модели. Вносим тестовые данные, проверяем правильность работы ограничений (нужно не менее 5 записей на таблицу).

Скрипт:

-- Создание таблицы "artist" (художники)

**CREATE** **TABLE** artist (

artist\_id SERIAL **PRIMARY** **KEY**,

artist\_name **VARCHAR**(100) **NOT** **NULL**,

country **VARCHAR**(50),

birth\_year **INT** **CHECK** (birth\_year **BETWEEN** 1000 **AND** **EXTRACT**(**YEAR** **FROM** **NOW**())),

death\_year **INT** **CHECK** (death\_year > birth\_year **OR** death\_year **IS** **NULL**)

);

-- Создание таблицы "style" (стили)

**CREATE** **TABLE** **style** (

style\_id SERIAL **PRIMARY** **KEY**,

style\_name **VARCHAR**(50) **NOT** **NULL** **UNIQUE**

);

-- Создание таблицы "artwork" (произведения искусства)

**CREATE** **TABLE** artwork (

artwork\_id SERIAL **PRIMARY** **KEY**,

title **VARCHAR**(200) **NOT** **NULL**,

year **INT** **NOT** **NULL** **CHECK** (year **BETWEEN** 1000 **AND** **EXTRACT**(**YEAR** **FROM** **NOW**())),

image\_path **VARCHAR**(255) **NOT** **NULL**,

artist\_id **INT** **NOT** **NULL** **REFERENCES** artist(artist\_id) **ON** **DELETE** **CASCADE**,

style\_id **INT** **NOT** **NULL** **REFERENCES** style(style\_id) **ON** **DELETE** **CASCADE**

);

-- Создание таблицы "exhibition" (выставки)

**CREATE** **TABLE** exhibition (

exhibition\_id SERIAL **PRIMARY** **KEY**,

exhibition\_name **VARCHAR**(100) **NOT** **NULL**,

description **TEXT**

);

-- Создание таблицы "exhibition\_artwork" (связь многие-ко-многим)

**CREATE** **TABLE** exhibition\_artwork (

exhibition\_id **INT** **NOT** **NULL** **REFERENCES** exhibition(exhibition\_id) **ON** **DELETE** **CASCADE**,

artwork\_id **INT** **NOT** **NULL** **REFERENCES** artwork(artwork\_id) **ON** **DELETE** **CASCADE**,

**PRIMARY** **KEY** (exhibition\_id, artwork\_id)

);

-- Создание таблицы "user" (пользователи)

**CREATE** **TABLE** user\_data (

user\_id SERIAL **PRIMARY** **KEY**,

username **VARCHAR**(50) **NOT** **NULL** **UNIQUE**,

email **VARCHAR**(100) **NOT** **NULL** **UNIQUE**,

user\_password **VARCHAR**(255) **NOT** **NULL**

);

-- Создание таблицы "favorite" (избранное)

**CREATE** **TABLE** favorite (

user\_id **INT** **NOT** **NULL** **REFERENCES** user\_data(user\_id) **ON** **DELETE** **CASCADE**,

artwork\_id **INT** **NOT** **NULL** **REFERENCES** artwork(artwork\_id) **ON** **DELETE** **CASCADE**,

**PRIMARY** **KEY** (user\_id, artwork\_id)

);

-- Создание таблицы "admin" (администраторы)

**CREATE** **TABLE** **admin** (

adminID SERIAL **PRIMARY** **KEY**,

login **VARCHAR**(50) **NOT** **NULL** **UNIQUE**,

adminPassword **VARCHAR**(255) **NOT** **NULL**

);

Внесем тестовые данные:

-- Добавление художников

**INSERT** **INTO** artist (artist\_name, country, birth\_year, death\_year) **VALUES**

('Клод Моне', 'Франция', 1840, 1926),

('Леонардо да Винчи', 'Италия', 1452, 1519),

('Фрида Кало', 'Мексика', 1907, 1954),

…

-- Добавление стилей

**INSERT** **INTO** **style** (style\_name) **VALUES**

('Импрессионизм'),

('Ренессанс'),

…

-- Добавление произведений

**INSERT** **INTO** artwork (title, **year**, image\_path, artist\_id, style\_id) **VALUES**

-- Импрессионизм

('Впечатление. Восход солнца', 1872, '/images/impression\_sunrise.jpg', 1, 1),

('Дама в саду Сент-Адресс', 1867, '/images/lady.jpg', 1, 1),

…

-- Ренессанс

('Мона Лиза', 1503, '/images/mona\_liza.jpg', 2, 2),

…

-- Добавление выставок

**INSERT** **INTO** exhibition (exhibition\_name, description) **VALUES**

('Импрессионизм', 'В попытке поймать ускользающее мгновение жизни'),

('Ренессанс', 'Шедевры эпохи Возрождения'),

('Современное искусство', 'Работы в ногу со временем');

-- Связь выставок и произведений

**INSERT** **INTO** exhibition\_artwork (exhibition\_id, artwork\_id)

**VALUES**

-- Импрессионизм

(

(**SELECT** e.exhibition\_id **FROM** exhibition e **WHERE** e.exhibition\_name = 'Импрессионизм'),

(**SELECT** a.artwork\_id **FROM** artwork a **WHERE** title = 'Впечатление. Восход солнца')

),

(

…

),

-- Ренессанс

(

(**SELECT** e.exhibition\_id **FROM** exhibition e **WHERE** e.exhibition\_name = 'Ренессанс'),

(**SELECT** a.artwork\_id **FROM** artwork a **WHERE** title = 'Мона Лиза')

),

(

(**SELECT** e.exhibition\_id **FROM** exhibition e **WHERE** e.exhibition\_name = 'Ренессанс'),

(**SELECT** a.artwork\_id **FROM** artwork a **WHERE** title = 'Рождение Венеры')

…

),

-- Современное искусство

(

(**SELECT** e.exhibition\_id **FROM** exhibition e **WHERE** e.exhibition\_name = 'Современное искусство'),

(**SELECT** a.artwork\_id **FROM** artwork a **WHERE** title = 'Две Фриды')

…

-- Добавление тестовых пользователей

**INSERT** **INTO** user\_data (username, email, user\_password) **values**

('karina', 'karina@mail.ru', '1234'),

…

-- Добавление администраторов

**INSERT** **INTO** admin (login, adminpassword) **VALUES**

('admin', '1234'),

…

-- Добавление произведений в избранное (пример)

**INSERT** **INTO** favorite (user\_id, artwork\_id) **VALUES**

(1, 1),

(1, 5),

(3, 3),

(4, 2),

(5, 4);

Проверим работу ограничений:

-- Попытка добавить дубликат стиля

**INSERT** **INTO** style (style\_name) **VALUES** ('Импрессионизм'); -- Ошибка: UNIQUE violation

-- Попытка некорректного года

**INSERT** **INTO** artwork (title, year, image\_path, artist\_id, style\_id)

**VALUES** ('Тест', 999, '/test.jpg', 1, 1); -- Ошибка: CHECK constraint

Этап 5

Проектируем и реализуем клиентское приложение. С помощью UML-диаграмм классов описываем структуру разрабатываемого приложения. Далее пишете приложение на Java. Используем Eclipse и работаем с базой данных через JDBC, не используя фреймворки и объектно-реляционные "оболочки".

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4 – Диаграмма классов

Реализация:

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, багет

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, пейзаж, багет

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как багет, текст, снимок экрана, женщина

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, Человеческое лицо, человек

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Этап 6

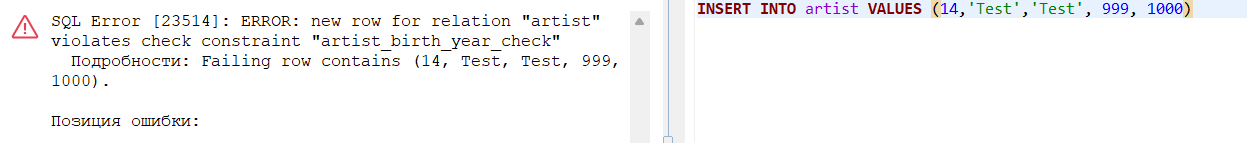
Вспоминаем, что вы знаете про тестирование ПО, описываем и реализуем план тестирования вашего приложения и базы данных.  
**Чек-лист для тестирования базы данных**

1. Общие проверки структуры БД:
   * Все таблицы (artist, style, artwork, exhibition, user, favorite) существуют ✅
2. Таблица Artist
   * Ограничение NOT NULL для artist.name ✅

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.

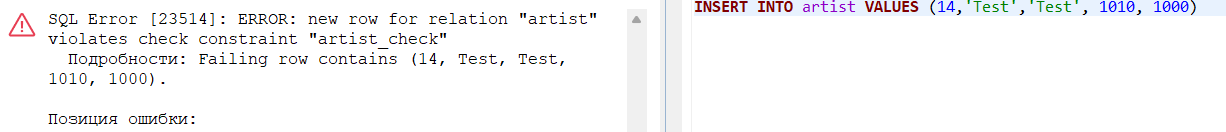
* + Проверка birth\_year (1000 ≤ year ≤ текущий) ✅



Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

* + Проверка death\_year > birth\_year ✅



1. Таблица artwork
   * Проверка е year BETWEEN 1000 AND текущий ✅

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

* + Каскадное удаление при удалении художника ✅

Изображение выглядит как текст, Шрифт, программное обеспечение, линия

Контент, сгенерированный ИИ, может содержать ошибки.

* + Ограничение image\_path NOT NULL ✅

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

1. Таблица exhibition\_artwork
   * Уникальность пары (exhibition\_id, artwork\_id) ✅



* + Каскадное удаление при удалении выставки ✅

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

1. Таблица user
   * Уникальность username и email ✅

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

* + Проверка длины пароля (user\_password) **❌**

Тест упал, добавлены ограничения:

**ALTER** **TABLE** "user"

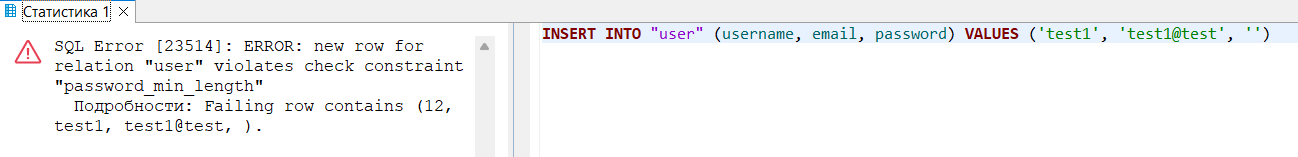
**ADD** **CONSTRAINT** password\_min\_length

**CHECK** (**LENGTH**(password) >= 4);

**ALTER** **TABLE** "admin"

**ADD** **CONSTRAINT** password\_min\_length

**CHECK** (**LENGTH**(password) >= 4);



**Чек-лист для тестирования клиентского приложения**

1. Основной функционал
   * Поиск по частичному названию ✅

Изображение выглядит как текст, снимок экрана, Человеческое лицо, Веб-сайт

Контент, сгенерированный ИИ, может содержать ошибки.

* + Сохранение работ в избранном для каждого авторизованного пользователя ✅

Изображение выглядит как текст, снимок экрана, млекопитающее, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

* + Изменение описания работ администратором ✅

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, Операционная система, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Этап 7

Попробуем сделать так, чтобы у нас было кроссплатформенное приложение. Т. е. оно работало бы на вашей основной ОС с выбранной СУБД, и на виртуальной машине с Ред ОС.

Для этого могут потребоваться какие-то конфигурационные файлы, прочитав из которых настройки, приложение будет использовать правильные параметры.

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.