

Федеральное государственное автономное образовательное учреждение
высшего образования
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа «Киберфизические системы и управление»

ОТЧЕТ ПО ПРОЕКТУ

по дисциплине: «Базы данных»

на тему: «Разработка Java-приложения, взаимодействующего с базой
данных»

Выполнила:
студент гр. 3532703/00101

<подпись>

Кимельфельд Д.П.

Преподаватель:

Доцент ВШ КФСнУ

<подпись>

Нестеров С.А.

Санкт-Петербург
2023

Оглавление

1. Доменная область	3
2. Требования	3
3. Проектирование ERD	4
2.1 Концептуальный уровень	4
2.2 Логический уровень	5
2.3 Физический уровень	6
4. Доказательство нормальных форм	7
4.1 Первая нормальная форма	7
4.2 Вторая нормальная форма	7
4.3 Нормальная форма Бойса-Кодда	8
5. Развертка базы данных	10
6. Тестовые данные	13
7. Клиент-приложение	15
8. Тестирование	16
8.1 Дымовое тестирование	16
8.2 Функциональное тестирование	18
8.3 Нефункциональное тестирование	18
Приложение	20

1. Доменная область

2. Требования

1. Создание генеалогического дерева осуществляется с помощью листьев лепестков, которые можно перемещать мышкой.
2. Добавление записей возможно, включая информацию о имени, фамилии и прочей текстовой информации.
3. Создание отношений между записями, таких как родительство, осуществляется с помощью лепестков.
4. Просмотр генеалогического дерева доступен в виде записей в таблице.
5. Редактирование записей осуществляется, включая изменение имени и фамилии.

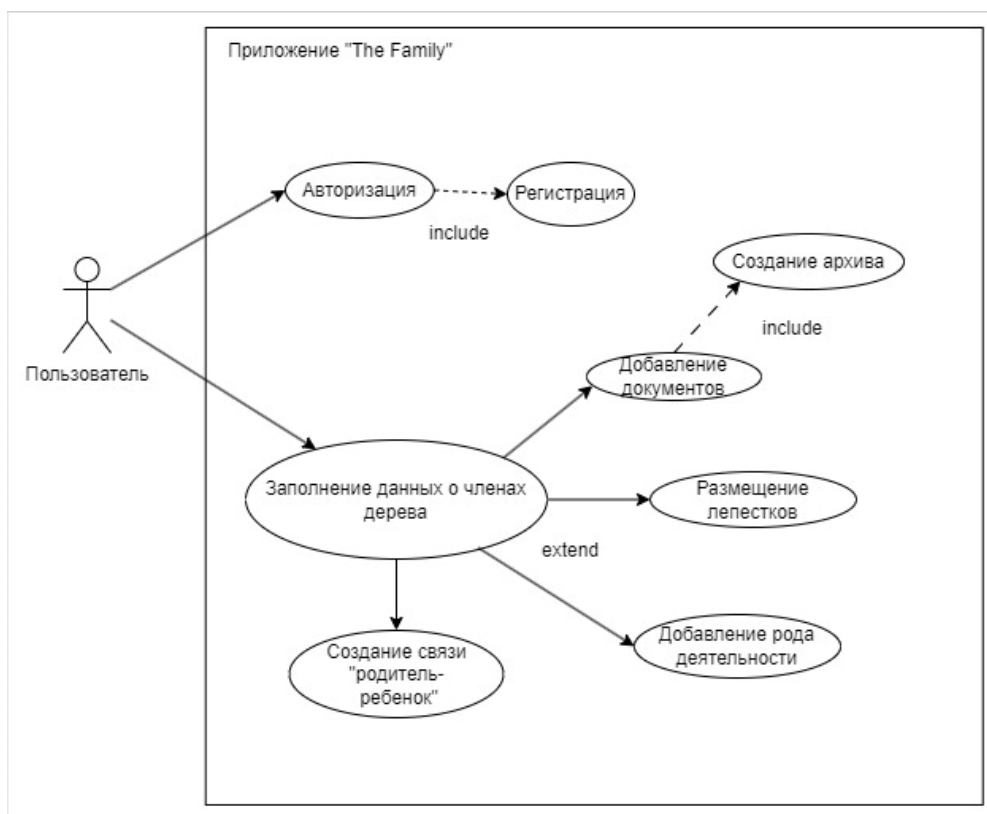


Рисунок 1.1 – Use-case диаграмма разрабатываемого приложения

6. Список событий, связанных с каждой записью, таких как дата рождения, дата смерти и другие, доступен для просмотра.
7. Создание, редактирование и удаление событий, связанных с записью, возможно.

8. Выгрузка скриншота приложения с изображением генеалогического дерева включена в функционал.
9. Доступ к приложению разрешен только одному пользователю.
10. Указание отношений между записями, соединяя лепестки на сцене прямыми линиями и указывая тип отношения в окошке, возможно.

3. Проектирование ERD

2.1 Концептуальный уровень

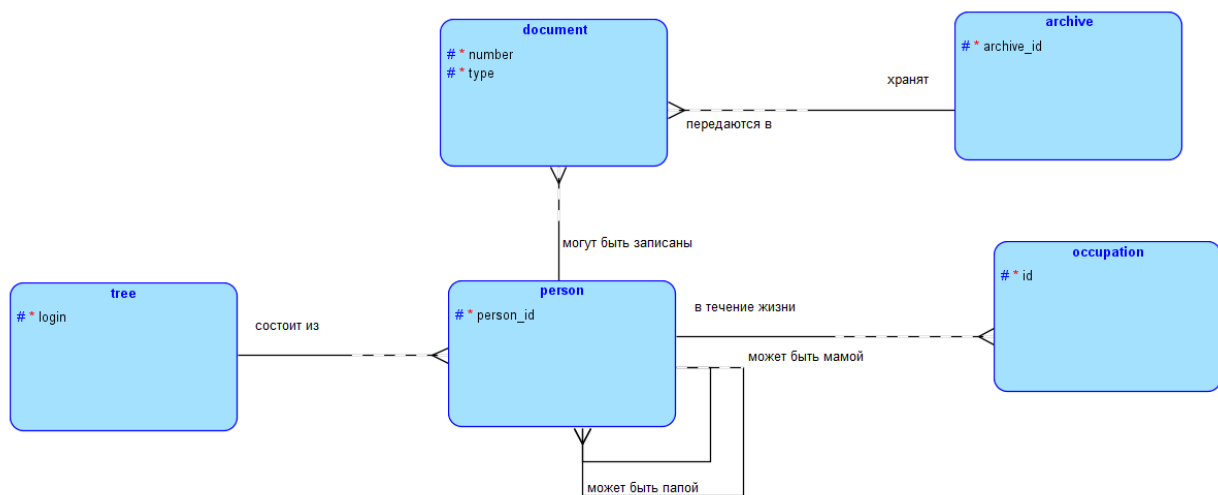


Рисунок 1 – Концептуальный уровень

2.2 Логический уровень

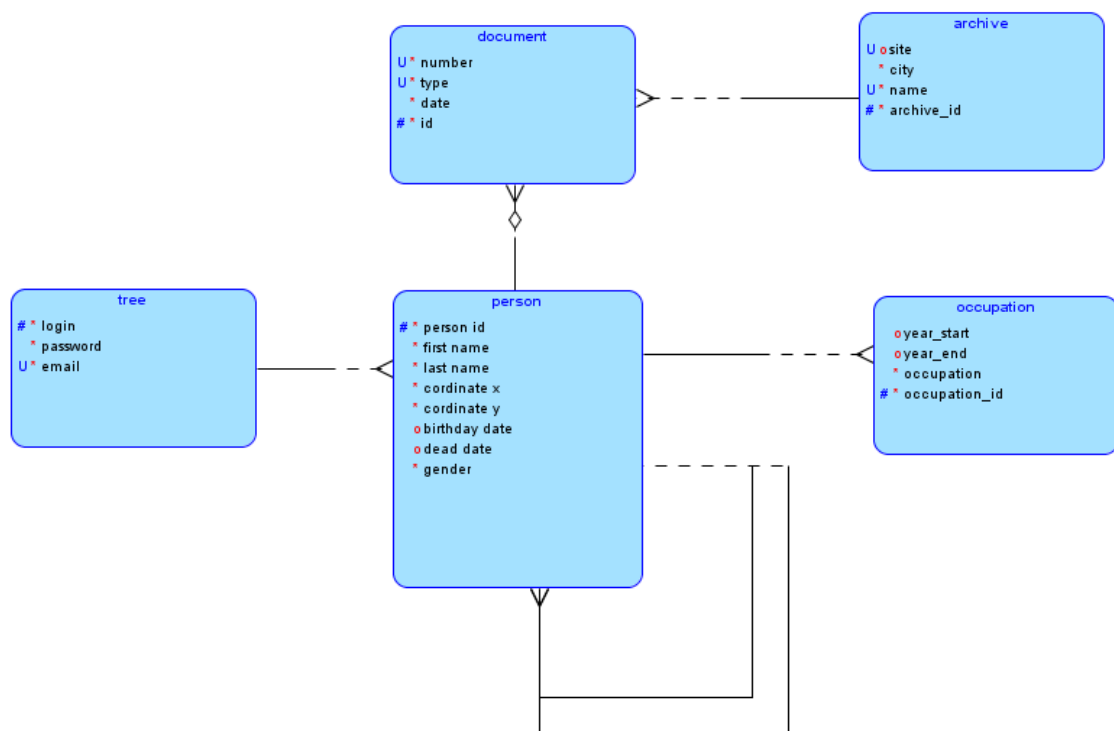


Рисунок 2 - Логический уровень

2.3 Физический уровень

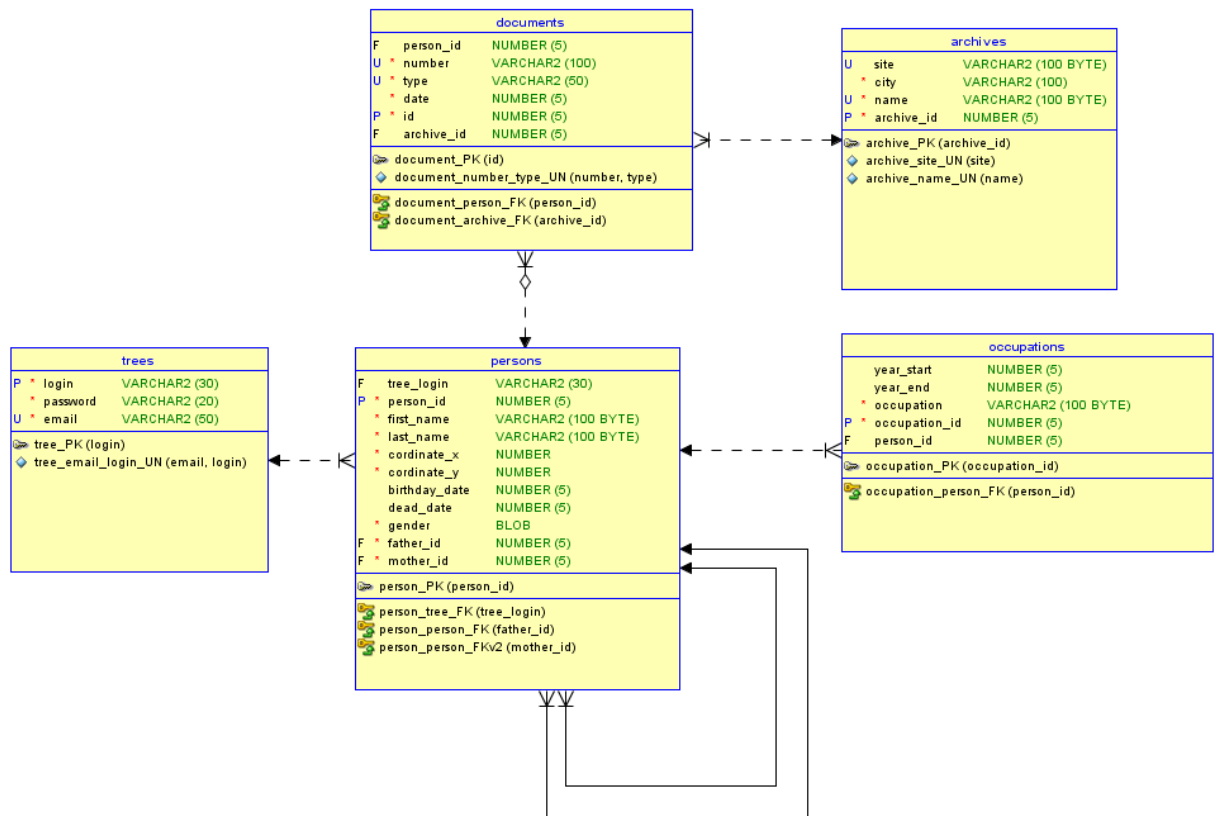


Рисунок 3 – Физический уровень

После того как была получена физическая модель данных необходимо доказать, что все таблицы находятся в нормальной форме Бойса-Кодда.

4. Доказательство нормальных форм

Определение. Отношение находится в **нормальной форме Бойса-Кодда**, тогда и только тогда, когда все детерминанты нетривиальных и неприводимых функциональных зависимостей являются потенциальными ключами и выполнены нормальные формы меньшего порядка.

4.1 Первая нормальная форма

Первая НФ требует, чтобы все первичные ключи были ненулевыми и значения скалярными.

Достаточно очевидно, что в сущностях, кроме trees, установлены суррогатные ненулевые ключи. В trees же логин, который также не может быть нулевым.

4.2 Вторая нормальная форма

Вторая НФ требует, чтобы была выполнена первая НФ и все атрибуты отношения, не входящие в первичный ключ, неприводимы зависимы от него.

Доказательство для таблицы “Trees”

У таблицы есть 2 потенциальных простых ключа, это логин и электронная почта. Они оба взаимозаменяемы и их ФЗ с остальными атрибутами будут неприводимо зависимыми.

Доказательство для таблицы “Persons”

Потенциальный ключ единственный и простой, таблица в 1 НФ, следовательно, вторая норма выполнена также.

Доказательство для таблицы “Documents”

Два потенциальных ключа – id и суперключ – (number, type). Касательно id все очевидно, ФЗ будет неприводимо зависимы.

Далее рассмотрим {number, type}:

$\{number, type\} \rightarrow \{date\}$

$\{\text{number, type}\} \rightarrow \{\text{archive_id}\}$

$\{\text{number, type}\} \rightarrow \{\text{person_id}\}$

$\{\text{number, type}\} \rightarrow \{\text{id}\}$

Очевидно, что ФЗ являются неприводимо зависимыми, так как невозможно убрать какой-то атрибут из суперключа без потери зависимости.

Доказательство для таблицы “Archives”

В таблице 3 потенциальных ключа: site, name, id. Рассмотрим site и name. Они оба взаимозаменяемы и их ФЗ с остальными атрибутами будут неприводимо зависимыми.

Суррогатный ключ по определению также имеет все неприводимо зависимые ФЗ с неключевыми атрибутами.

Доказательство для таблицы “Occupations”

Ключ простой и единственный, выполнена 1 НФ, следовательно, автоматически в 2 НФ.

4.3 Нормальная форма Бойса-Кодда

Нетривиальная неприводимая зависимость возникает, когда существует зависимость между двумя или более наборами атрибутов, которую нельзя разбить на более мелкие функциональные зависимости.

Неприводимая зависимость означает, что ни одно подмножество атрибутов в зависимости не может быть удалено без нарушения смысла зависимости.

Доказательство для таблицы “Trees”

$\{\text{login}\} \rightarrow \{\text{email}\}$

$\{\text{login}\} \rightarrow \{\text{password}\}$

$\{\text{email}\} \rightarrow \{\text{password}\}$

$\{\text{email}\} \rightarrow \{\text{login}\}$

Все детерминанты являются потенциальными ключами.

Доказательство для таблицы “Persons”

$\{\text{person_id}\} \rightarrow \{\text{first_name}\}$

$\{\text{person_id}\} \rightarrow \{\text{tree_login}\}$

$\{\text{person_id}\} \rightarrow \{\text{birthday_year}\}$

...

Все детерминанты являются потенциальными ключами.

Доказательство для таблицы “Documents”

$\{\text{number, type}\} \rightarrow \{\text{date}\}$

$\{\text{number, type}\} \rightarrow \{\text{archive_id}\}$

$\{\text{number, type}\} \rightarrow \{\text{person_id}\}$

$\{\text{number, type}\} \rightarrow \{\text{id}\}$

$\{\text{id}\} \rightarrow \{\text{person_id}\}$

$\{\text{id}\} \rightarrow \{\text{number, type}\}$

$\{\text{id}\} \rightarrow \{\text{archive_id}\}$

$\{\text{id}\} \rightarrow \{\text{date}\}$

Все детерминанты являются потенциальными ключами.

Доказательство для таблицы “Archives”

Все потенциальные ключи простые, транзитивных зависимостей нет – нормальная форма Бойса-Кода, так как все детерминанты – потенциальные ключи в нетривиальных и неприводимых зависимостях.

Доказательство для таблицы “Occupations”

Все потенциальные ключи простые, транзитивных зависимостей нет – нормальная форма Бойса-Кода, так как все детерминанты – потенциальные ключи в нетривиальных и неприводимых зависимостях.

5. Развертка базы данных

Скрипт был запущен, таблицы создались. В качестве СУБД был выбран Oracle Database.

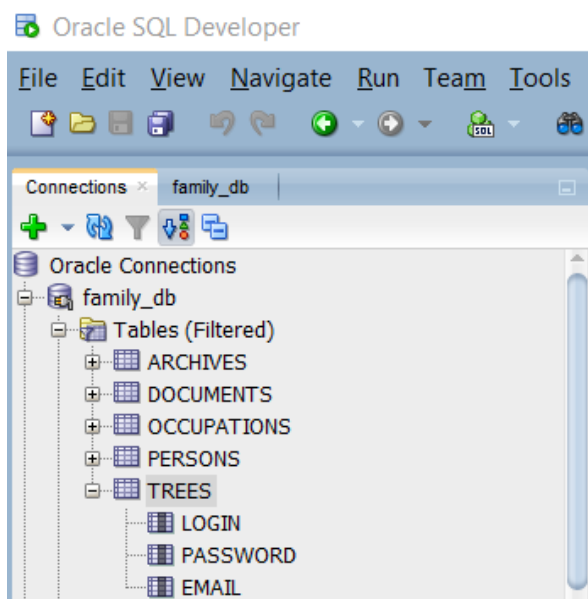


Рисунок 5.1 – Именованние таблиц базы данных

По ходу работы с клиентом были внесены изменения относительно генерального скрипта, касающиеся названий столбцов и ограничений. Ниже можно ознакомиться с конечным видом таблиц.

Columns						
Columns Data Model Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL						
Actions...						
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	SITE	VARCHAR2(100 BYTE)	Yes	(null)	1 (null)	
2	CITY	VARCHAR2(100 BYTE)	No	(null)	2 (null)	
3	NAME	VARCHAR2(100 BYTE)	No	(null)	3 (null)	
4	ARCHIVE_ID	NUMBER(38,0)	No	"FAMILY_DB...	4 (null)	

Рисунок 5.2 – Таблица Archives

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 PERSON_ID	NUMBER(38,0)	Yes	(null)	1	(null)
2 NUMBER_DOC	VARCHAR2(100 BYTE)	No	(null)	2	(null)
3 TYPE	VARCHAR2(50 BYTE)	No	(null)	3	(null)
4 YEAR	NUMBER(5,0)	No	(null)	4	(null)
5 ID	NUMBER(38,0)	No	"FAMILY_DB...	5	(null)
6 ARCHIVE_ID	NUMBER(38,0)	Yes	(null)	6	(null)

Рисунок 5.3 – Таблица Documents

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 YEAR_START	NUMBER(5,0)	Yes	(null)	1	(null)
2 YEAR_END	NUMBER(5,0)	Yes	(null)	2	(null)
3 OCCUPATION	VARCHAR2(100 BYTE)	No	(null)	3	(null)
4 OCCUPATION_ID	NUMBER(38,0)	No	"FAMILY_DB...	4	(null)
5 PERSON_ID	NUMBER(38,0)	Yes	(null)	5	(null)

Рисунок 5.4 – Таблица Occupations

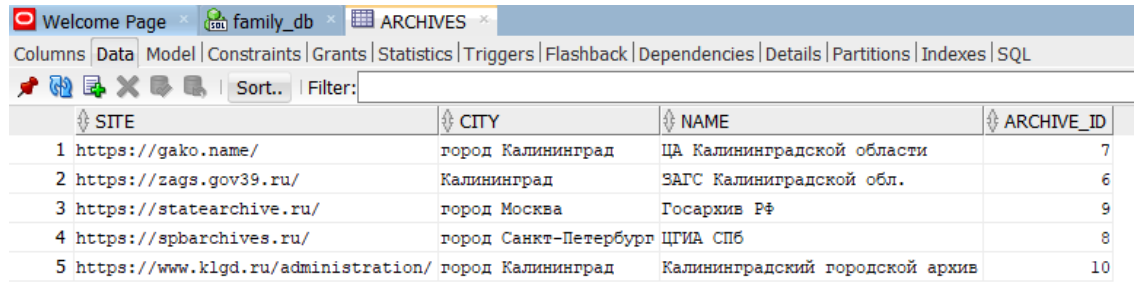
COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 TREE_LOGIN	VARCHAR2(30 BYTE)	No	(null)	1	(null)
2 PERSON_ID	NUMBER(38,0)	No	"FAMILY_DB...	2	(null)
3 FIRST_NAME	VARCHAR2(100 BYTE)	No	(null)	3	(null)
4 LAST_NAME	VARCHAR2(100 BYTE)	No	(null)	4	(null)
5 COORDINATE_X	NUMBER	Yes	(null)	5	(null)
6 COORDINATE_Y	NUMBER	Yes	(null)	6	(null)
7 BIRTHDAY_DATE	NUMBER(5,0)	Yes	(null)	7	(null)
8 DEAD_DATE	NUMBER(5,0)	Yes	(null)	8	(null)
9 GENDER	CHAR(1 BYTE)	No	(null)	9	(null)
10 FATHER_ID	NUMBER(38,0)	Yes	(null)	10	(null)
11 MOTHER_ID	NUMBER(38,0)	Yes	(null)	11	(null)

Рисунок 5.5 – Таблица Persons

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	LOGIN	VARCHAR2 (30 BYTE)	No	(null)	1 (null)	
2	PASSWORD	VARCHAR2 (20 BYTE)	No	(null)	2 (null)	
3	EMAIL	VARCHAR2 (50 BYTE)	No	(null)	3 (null)	

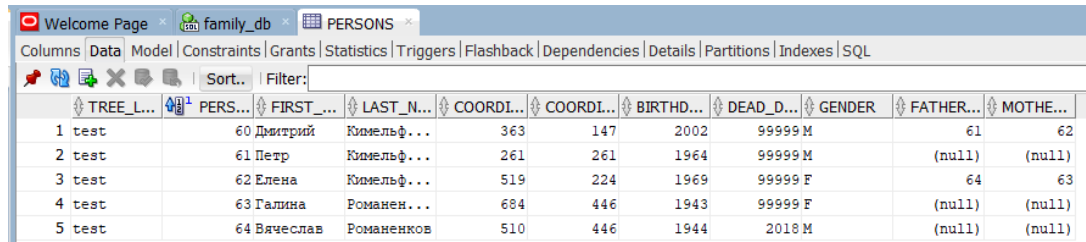
Рисунок 5.6 – Таблица Trees

6. Тестовые данные



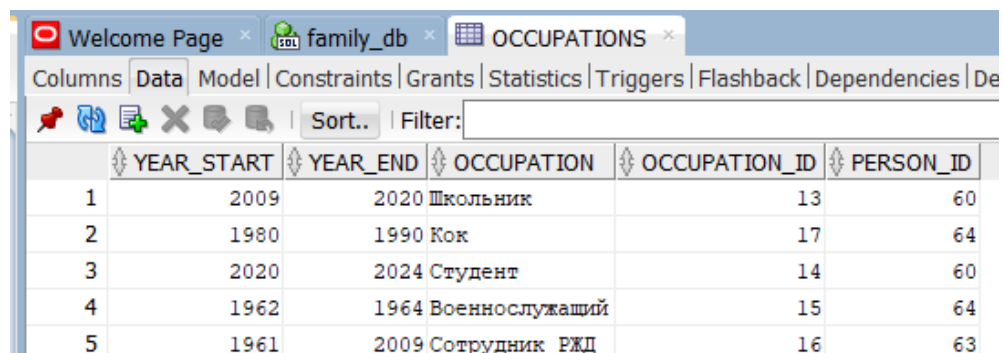
	SITE	CITY	NAME	ARCHIVE_ID
1	https://gako.name/	город Калининград	ЦА Калининградской области	7
2	https://zags.gov39.ru/	Калининград	ЗАГС Калининградской обл.	6
3	https://statearchive.ru/	город Москва	Госархив РФ	9
4	https://spbarchives.ru/	город Санкт-Петербург	ЦГИА СПб	8
5	https://www.klgd.ru/administration/	город Калининград	Калининградский городской архив	10

Рисунок 6.1 – данные для таблицы Archives



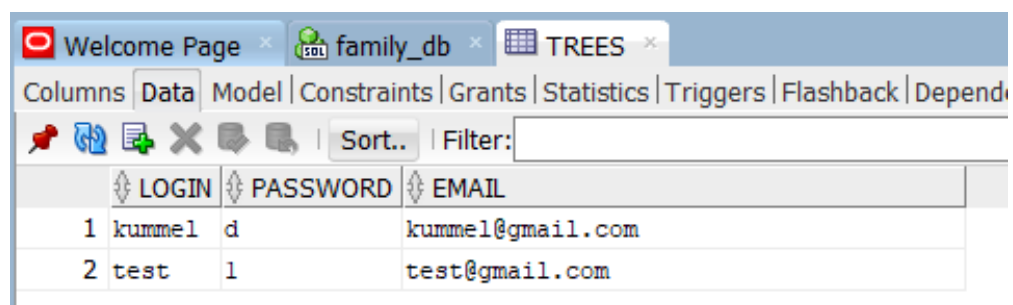
	TREE_L...	PERS...	FIRST...	LAST_N...	COORDI...	COORDI...	BIRTHD...	DEAD_D...	GENDER	FATHER...	MOTHE...
1	test	60	Дмитрий	Кумельф...	363	147	2002	99999	M	61	62
2	test	61	Петр	Кумельф...	261	261	1964	99999	M	(null)	(null)
3	test	62	Елена	Кумельф...	519	224	1969	99999	F	64	63
4	test	63	Галина	Романен...	684	446	1943	99999	F	(null)	(null)
5	test	64	Вячеслав	Романенков	510	446	1944	2018	M	(null)	(null)

Рисунок 6.2 – данные для таблицы Persons



	YEAR_START	YEAR_END	OCCUPATION	OCCUPATION_ID	PERSON_ID
1	2009	2020	Школьник	13	60
2	1980	1990	Кок	17	64
3	2020	2024	Студент	14	60
4	1962	1964	Военнослужащий	15	64
5	1961	2009	Сотрудник РЖД	16	63

Рисунок 6.3 – данные для таблицы Occupations



	LOGIN	PASSWORD	EMAIL
1	kummel	d	kummel@gmail.com
2	test	1	test@gmail.com

Рисунок 6.4 – данные для таблицы Trees

	PERSON_ID	NUMBER_DOC	TYPE	YEAR	ID	ARCHIVE_ID
1	64	67891	Медицинская карта	2000	22	8
2	62	77734	Паспорт	2000	23	8
3	60	123	Свидетельство о рождении	2002	20	7

Рисунок 6.5 – данные для таблицы Documents

Таким образом, можно перейти к разработке клиента под базу данных, с помощью которого можно будет взаимодействовать с СУБД.

7. Клиент-приложение

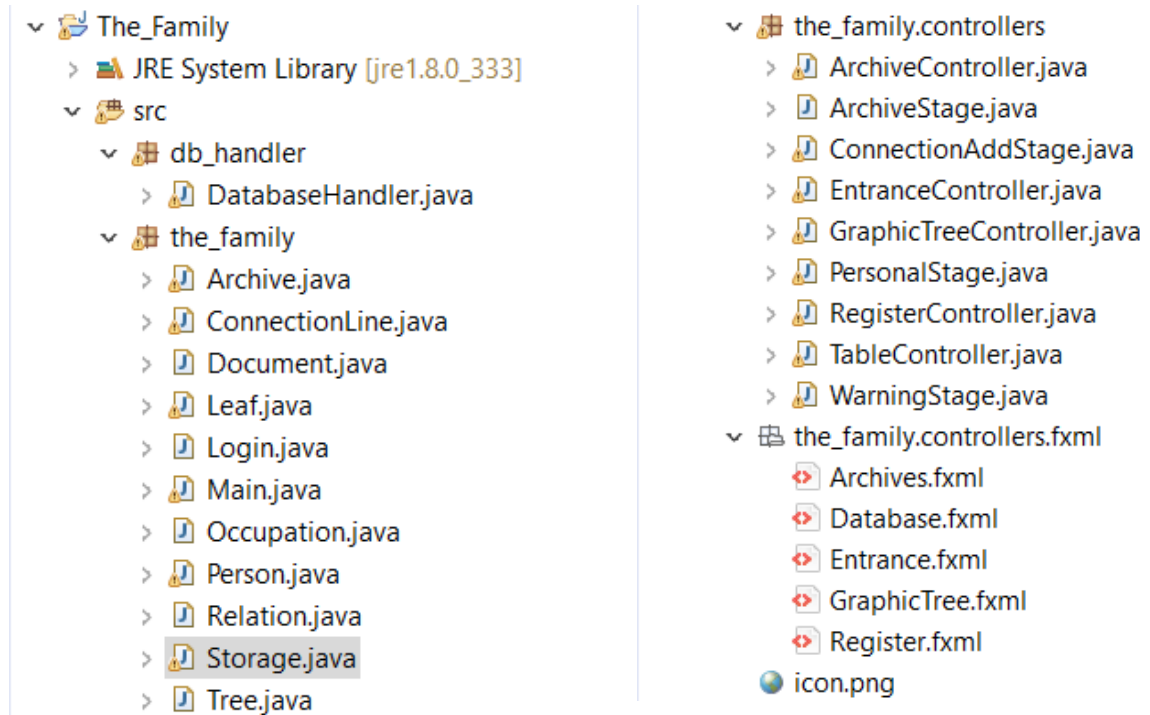


Рисунок 7.1 – Иерархия классов и пакетов в проекте

В приложение №2 можно ознакомиться с диаграммой классов.

8. Тестирование

8.1 Дымовое тестирование

Суть дымового тестирования заключается в проверке готовности приложения к основному тестированию. Оно представляет собой короткий цикл тестов, подтверждающий, что приложение успешно запускается и выполняет основные функции.

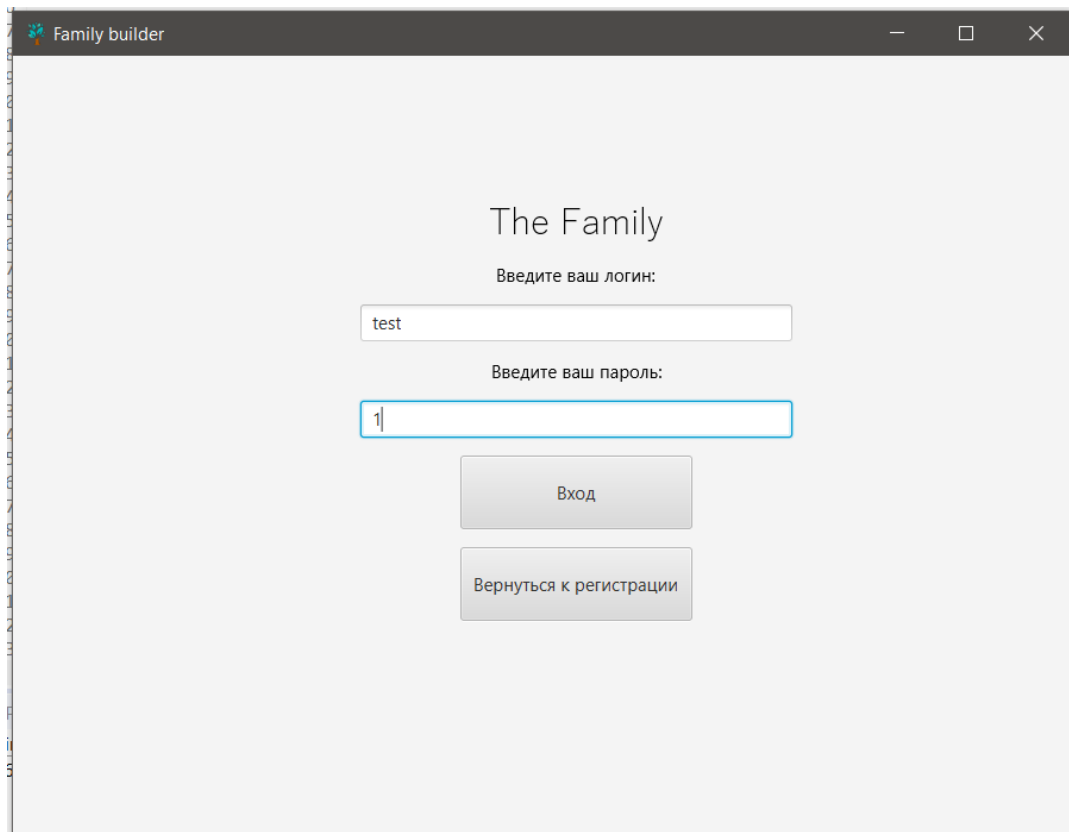


Рисунок 8.1 – Авторизация

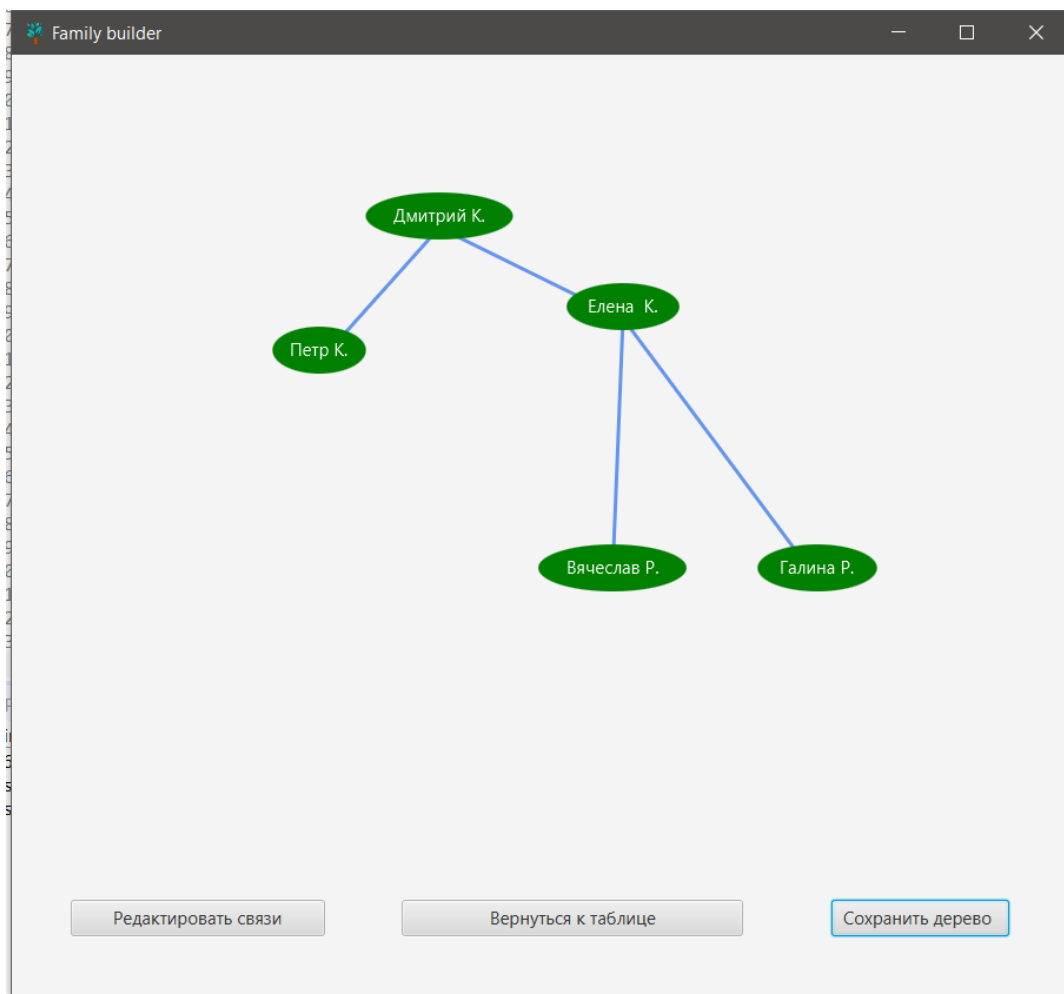


Рисунок 8.2 – Отображение дерева

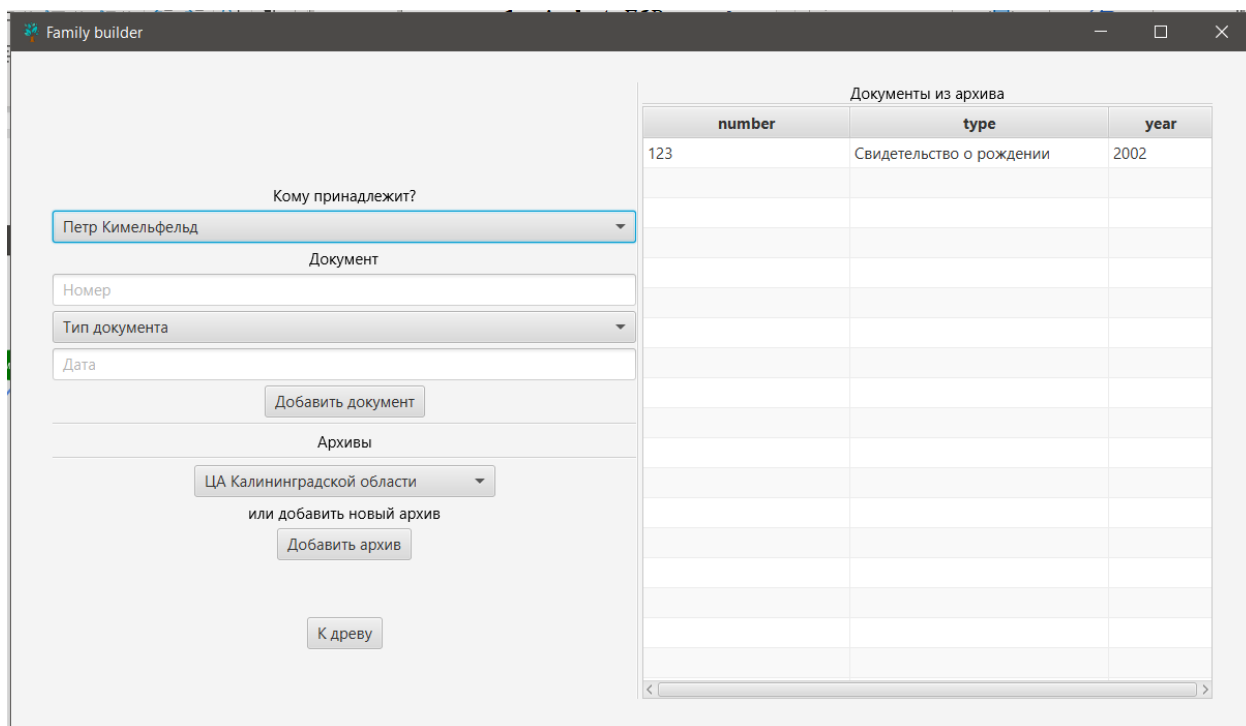


Рисунок 8.3 – Документы и архивы

8.2 Сессионное тестирование

Алгоритм сессионного тестирования:

1. Определить конкретную цель
2. Выделить ограниченный промежуток времени на исследование
3. Проверить целевую функциональность в течение выделенного времени

Выберем цель: проверить как сохраняются родители между авторизациями.

8.3 Позитивные и негативные тесты

Позитивные тесты заключаются в проверке того, что программа делает то, что должна и работает правильно.

Негативные тесты заключаются в проверке ввода некорректных данных и работе с ними.

Сделаем 2 теста.

1) Введем свидетельство рождения или свидетельство смерти о персоне, которое не совпадет со сведениями о человеке

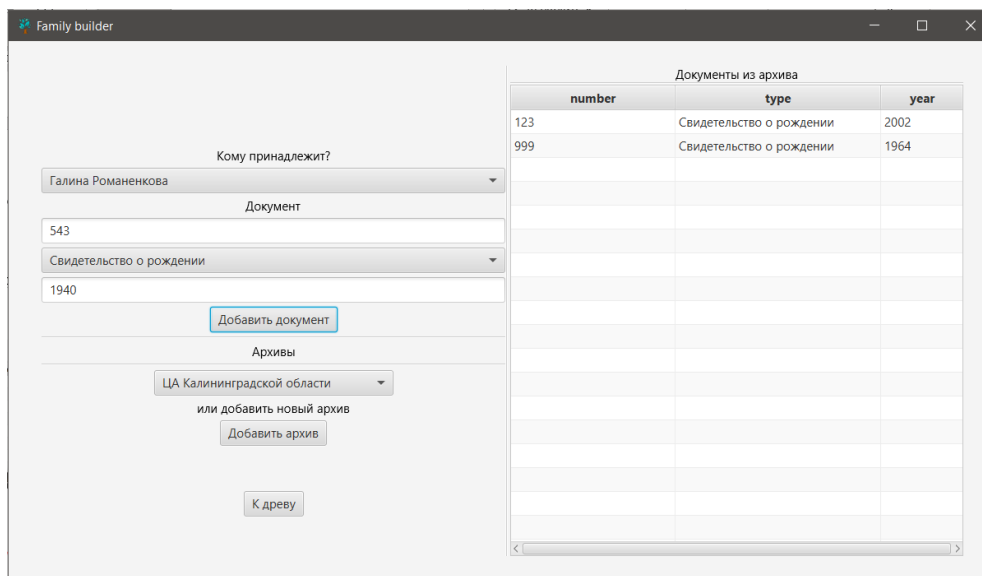


Рисунок 8.4 – Несовпадение документа с данными

В данном кейсе происходит проверка данных с тем, что указано о человеке. Оно не позволяет заполнить документ о персоне.

2) Создадим существующий документ, который уже есть в БД либо принадлежит другому человеку

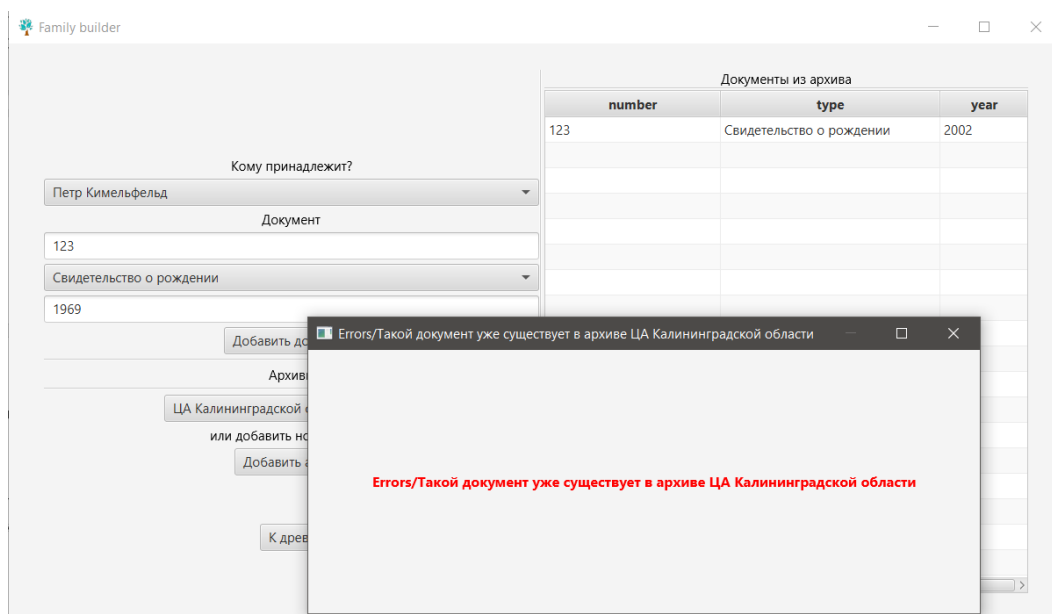


Рисунок 8.5 – Ошибка, предупреждающая об существующем документе

Приложение

Приложение № 1 – Скрипт для создания таблиц в Oracle Database

```
CREATE TABLE archives (  
    site      VARCHAR2(100 BYTE),  
    city      VARCHAR2(100) NOT NULL,  
    name      VARCHAR2(100 BYTE) NOT NULL,  
    archive_id NUMBER(5) NOT NULL  
);  
  
ALTER TABLE archives ADD CONSTRAINT archive_pk PRIMARY KEY ( archive_id );  
ALTER TABLE archives ADD CONSTRAINT archive_site_un UNIQUE ( site );  
ALTER TABLE archives ADD CONSTRAINT archive_name_un UNIQUE ( name );  
  
CREATE TABLE documents (  
    person_id  NUMBER(5),  
    number_doc VARCHAR2(100) NOT NULL,  
    type       VARCHAR2(50) NOT NULL,  
    year       NUMBER(5) NOT NULL,  
    id         NUMBER(5) NOT NULL,  
    archive_id NUMBER(5)  
);  
  
ALTER TABLE documents ADD CONSTRAINT document_pk PRIMARY KEY ( id );  
ALTER TABLE documents ADD CONSTRAINT document_number_type_un UNIQUE ( number_doc,  
                                                                    type );  
  
CREATE TABLE occupations (  
    year_start  NUMBER(5),  
    year_end    NUMBER(5),  
    occupation   VARCHAR2(100 BYTE) NOT NULL,  
    occupation_id NUMBER(5) NOT NULL,  
    person_id    NUMBER(5)  
);  
  
ALTER TABLE occupations ADD CONSTRAINT occupation_pk PRIMARY KEY ( occupation_id );  
  
CREATE TABLE persons (  
    tree_login VARCHAR2(30),
```

```

    person_id    NUMBER(5) NOT NULL,
    first_name    VARCHAR2(100 BYTE) NOT NULL,
    last_name     VARCHAR2(100 BYTE) NOT NULL,
    cordinate_x   NUMBER NOT NULL,
    cordinate_y   NUMBER NOT NULL,
    birthday_date NUMBER(5),
    dead_date     NUMBER(5),
    gender        BLOB NOT NULL,
    father_id     NUMBER(5),
    mother_id     NUMBER(5)
);
ALTER TABLE persons ADD CONSTRAINT person_pk PRIMARY KEY ( person_id );
CREATE TABLE trees (
    login    VARCHAR2(30) NOT NULL,
    password VARCHAR2(20) NOT NULL,
    email    VARCHAR2(50) NOT NULL
);
ALTER TABLE trees ADD CONSTRAINT tree_pk PRIMARY KEY ( login );
ALTER TABLE trees ADD CONSTRAINT tree_email_login_un UNIQUE ( email,
                                login );

ALTER TABLE documents
    ADD CONSTRAINT document_archive_fk FOREIGN KEY ( archive_id )
        REFERENCES archives ( archive_id );
ALTER TABLE documents
    ADD CONSTRAINT document_person_fk FOREIGN KEY ( person_id )
        REFERENCES persons ( person_id );
ALTER TABLE occupations
    ADD CONSTRAINT occupation_person_fk FOREIGN KEY ( person_id )
        REFERENCES persons ( person_id );
ALTER TABLE persons
    ADD CONSTRAINT person_person_fk FOREIGN KEY ( father_id )
        REFERENCES persons ( person_id );
ALTER TABLE persons

```

```

ADD CONSTRAINT person_person_fkv2 FOREIGN KEY ( mother_id )
    REFERENCES persons ( person_id );
ALTER TABLE persons
    ADD CONSTRAINT person_tree_fk FOREIGN KEY ( tree_login )
    REFERENCES trees ( login );
CREATE OR REPLACE TRIGGER fknto_documents BEFORE
    UPDATE OF person_id ON documents
    FOR EACH ROW
BEGIN
    IF :old.person_id IS NOT NULL THEN
        raise_application_error(-20225, 'Non Transferable FK constraint document_person_FK on
table documents is violated');
    END IF;
END;

```

Приложение №2 – Диаграмма классов

