Проект върху тема:

Задание 2: Университетска информационна система

По дисциплината:

Информационни системи

Изготвил: Михаил Христов

Специалност: Софтуерно инженерство

Фак. Номер: 16

Contents

1. УВОД

1.1. Актуалност на проблема

В началото на XXI век дигиталната трансформация обхваща всички сфери на обществения живот, като образованието не прави изключение. В съвременните висши учебни заведения обработката на информационни потоци е не просто административна дейност, а критичен бизнес процес, от който зависи качеството на обучение и ефективността на академичния състав. Ежедневното функциониране на един университет генерира огромни масиви от хетерогенни данни – лични досиета на студенти, академична заетост на преподаватели, учебни планове, разписания на дисциплини и, най-важното, резултати от изпити (оценки).

Управлението на тези данни изисква надеждни, сигурни и автоматизирани средства. Въпреки навлизането на новите технологии, в много по-малки звена, катедри или факултети все още се наблюдават остарели практики. Традиционните методи за водене на документация на хартиен носител или използването на неструктурирани файлови формати (като текстови документи или разпокъсани електронни таблици в Excel) крият сериозни рискове.

1.2. Дефиниране на проблемите при традиционния подход

Основните недостатъци на неавтоматизираните системи могат да се систематизират в следните направления:

1. **Нарушаване на целостта на данните (Data Integrity):** При ръчно въвеждане на информация вероятността за човешка грешка е висока.
2. **Излишество на данни (Data Redundancy):** Една и съща информация (например име на студент) често се записва на множество места, което води до несъответствия при промяна.
3. **Затруднени справки:** Извличането на обобщена информация (напр. "Среден успех на всички студенти по специалност КСТ") от хартиени дневници е бавен и трудоемък процес.
4. **Липса на защита:** Хартиените носители и локалните файлове лесно могат да бъдат изгубени или унищожени.

1.3. Цел на разработката

Проектът цели да дигитализира и оптимизира административните процеси чрез внедряване на съвременни софтуерни подходи. Основната идея е замяната на плоските файлове с **релационна база данни**, която гарантира консистентност, и предоставянето на **интуитивен графичен потребителски интерфейс (GUI)**, който да улесни работата на крайния потребител (администратора), без да изисква от него задълбочени технически познания.

2. ЛИТЕРАТУРЕН ПРЕГЛЕД И АНАЛИЗ НА ПРОБЛЕМА

2.1. Анализ на съществуващите решения

Пазарът на софтуер за образователни институции предлага множество решения, но те често са поляризирани в две крайности. От една страна са мащабните системи за управление на обучението (LMS - Learning Management Systems) и ERP системите за университети (като Blackboard, Moodle, или специализирани университетски ERP).

- **Предимства:** Те са мощни, уеб-базирани и многофункционални.
- **Недостатъци:** Тези системи са изключително скъпи за лицензиране и поддръжка. Внедряването им изисква скъпоструваща сървърна инфраструктура, постоянна интернет свързаност и екип от системни администратори за поддръжка. За нуждите на малка катедра или за учебни цели, те са „тежки“ и икономически необосновани.

От другата страна са „направи-си-сам“ решенията, базирани на Microsoft Excel или Access. Въпреки че са достъпни, те не предлагат нужната валидация на данните и не поддържат сложни релационни връзки, което води до т.нар. „аномалии“ в базата данни (anomalies in database design).

Анализът на проблема показва ясна пазарна ниша и нужда от **леки (lightweight) клиентски приложения**. Този тип софтуер не изисква инсталация на тежки сървъри, работи локално на машината на потребителя и предоставя мигновена реакция на интерфейса.

2.2. Обосновка на избора на технологии

За реализацията на поставената задача е избрана технологичната комбинация от езика за програмиране **Python**, системата за управление на бази данни **SQLite** и библиотеката за графичен интерфейс **Tkinter**. Този избор се налага като индустриален стандарт за бърза разработка на десктоп приложения (RAD - Rapid Application Development) поради следните причини:

2.2.1. Език за програмиране Python

Python е интерпретиран език от високо ниво, който през последните години се утвърди като лидер в софтуерната индустрия.

- **Скорост на разработка:** Синтаксисът на Python е чист и четим, което позволява на програмиста да се фокусира върху бизнес логиката, а не върху сложните декларации, характерни за езици като C++ или Java.
- **Богата стандартна библиотека:** Концепцията „Batteries Included“ означава, че Python идва с вградени модули за работа с бази данни (sqlite3) и графични интерфейси (tkinter), което елиминира нуждата от сложни инсталации на външни зависимости.

2.2.2. Система за управление на бази данни SQLite

За разлика от клиент-сървър системите като MySQL или PostgreSQL, **SQLite** е вградена (embedded) база данни.

- **Архитектура:** Цялата база данни се съхранява в един-единствен файл на диска. Това прави преместването на системата (deployment) изключително лесно – просто копиране на файла.
- **Zero-Configuration:** Не изисква инсталация на сървър, създаване на потребители или настройки на портове.
- **Релационен модел:** Въпреки компактността си, SQLite поддържа пълния SQL стандарт, включително транзакции (ACID compliance), външни ключове (Foreign Keys) и сложни заявки (JOINs), което е задължително условие за настоящия проект.

2.2.3. Графичен потребителски интерфейс (GUI) – Tkinter и ttkbootstrap

Създаването на конзолно приложение не е удачно за крайни потребители. Ето защо е избран **Tkinter** – стандартният GUI инструментариум на Python.

- **Крос-платформена съвместимост:** Приложението ще изглежда и работи еднакво добре както на Windows, така и на Linux и macOS.
- **Event-Driven Programming:** Tkinter използва модел, управляван от събития (натискане на бутон, избор от меню), което е естественият начин за взаимодействие с потребителя.
- **Модерен дизайн:** За да се избегне остарелият вид на стандартния Tkinter, в проекта се използва библиотеката **ttkbootstrap**. Тя надгражда стандартните компоненти с модерни теми (като използваната в проекта тема "Superhero"), предоставяйки професионален външен вид (UI/UX), сравним със съвременните уеб приложения.

3. ЦЕЛ И ЗАДАЧИ НА ПРОЕКТА

3.1. Основна цел

Основната цел на настоящия курсов проект е проектирането и пълната софтуерна реализация на „Академична Информационна Система“ (АИС). Системата е предназначена да обслужва административните нужди на университетска структура (факултет или катедра), като предоставя единна централизирана среда за управление на учебния процес.

Ключов аспект на целта е не само създаването на потребителски интерфейс, а изграждането на стабилна архитектура, която да **поддържа целостта на данните (Data Integrity)**. Това означава системата да не допуска логически противоречия – например съществуване на оценки за несъществуващи студенти или дублиране на уникални идентификатори. В допълнение, системата цели да минимизира времето за административно обслужване чрез предоставяне на интуитивен графичен интерфейс, достъпен за служители без специализирано IT образование.

3.2. Конкретни задачи

За успешното постигане на дефинираната цел, разработката е декомпозирана на следните конкретни задачи:

1. **Проектиране на релационен модел на базата данни (Entity-Relationship Model):**
    - Анализ на предметната област и идентифициране на основните същности (Entities) – Студенти, Преподаватели, Дисциплини.
    - Определяне на атрибутите на всяка същност и типовете данни.
    - Установяване на връзките между тях (One-to-Many, Many-to-Many) и нормализация на базата данни до Трета нормална форма (3NF) за избягване на аномалии при вмъкване, обновяване и изтриване.
2. **Физическа реализация на базата данни чрез SQLite3:**
    - Написване на DDL (Data Definition Language) скриптове за създаване на таблиците.
    - Реализиране на механизма за **референтен интегритет** чрез външни ключове (Foreign Keys).
    - Конфигуриране на правила за каскадно изтриване (`ON DELETE CASCADE`), което гарантира, че при премахване на запис (напр. студент), свързаните с него данни (оценки) също ще бъдат коректно почистени.
3. **Разработка на Графичен Потребителски Интерфейс (GUI):**
    - Създаване на основен прозорец с навигационна структура тип „Tabbed Interface“ (табове) за логическо разделение на модулите.
    - Проектиране на входни форми (Forms) с подходящи контролни елементи – текстови полета (`Entry`) за свободно писане и падащи списъци (`Combobox`) за избор на релационни данни, което минимизира грешките при въвеждане.

4. **Имплементация на CRUD операции:**
    - Реализиране на пълния жизнен цикъл на данните:
        - **Create (Създаване):** Добавяне на нови записи в базата.
        - **Read (Четене):** Извличане и визуализация на данни в табличен вид (Treeview).
        - **Update (Обновяване):** (Включено като възможност за разширение).
        - **Delete (Изтриване):** Премахване на неактуална информация.
5. **Реализация на бизнес логика и валидация:**
    - Вграждане на алгоритми за обработка на данните на ниво приложение (Application Layer).
    - Автоматично изчисляване на Среден успех (GPA) чрез агрегиращи SQL функции (AVG).
    - Валидация на входните данни (Input Validation) – проверка дали оценките са в допустимия диапазон (2.00–6.00) и дали задължителните полета са попълнени преди изпращане на заявка към базата.

4. МАТЕРИАЛИ И МЕТОДИ

4.1. Използвани материали и технологии

Изборът на технологичен стек е продиктуван от изискванията за бърза разработка, преносимост и надеждност.

- Език за програмиране – Python 3.10+:

  Използван е езикът Python поради неговия синтаксис от високо ниво и динамична типизация. Версия 3.10+ предоставя подобрени възможности за обработка на грешки и оптимизация на работата с паметта. Богатата стандартна библиотека на Python позволява разработката да се случи без външни зависимости за основните функционалности.

- Среда за разработка (IDE) – Visual Studio Code:

  За написването на кода е използван Visual Studio Code (VS Code). Изборът е обоснован от наличието на мощни разширения за Python (Pylance), вграден дебъгер (Debugger), поддръжка на Git контрол на версиите и възможност за преглед на SQL файлове директно в средата.

- Система за управление на бази данни – SQLite3:

  SQLite е вградена библиотека, която реализира самостоятелна, безсървърна (serverless) SQL база данни.

- **Графична библиотека – Tkinter и ttkbootstrap:**
    - **Tkinter:** Това е стандартният Python интерфейс към GUI инструментариума Tk. Той е лек и идва инсталиран с Python.
    - **ttkbootstrap:** Тъй като стандартният Tkinter има остарял визуален стил (подобен на Windows 95), е използвана надстройката ttkbootstrap. Тя прилага модерни теми, вдъхновени от Bootstrap (в конкретния проект е използвана темата **"Superhero"**), и предоставя подобрени уиджети с професионален дизайн.

4.2. Източници на информация

При проектирането и реализацията са използвани следните основни източници:

1. **Официална документация на Python (docs.python.org):** За справки относно синтаксиса, модула sqlite3 и класовете на tkinter.
2. **Спецификации на SQL стандарта:** За написването на коректни заявки за създаване на таблици (CREATE TABLE), манипулация (INSERT, DELETE) и извличане (SELECT, JOIN).
3. **Ttkbootstrap Documentation:** За настройка на визуалните стилове и темите.
4. **Учебни материали:** Лекционни записки по дисциплините "Бази данни" и "Обектно-ориентирано програмиране".

4.3. Описание на заданието и предметната област

Системата моделира дейността на учебен отдел, като обхваща следните информационни обекти:

- **Студенти:** Основна единица в системата. За всеки студент се съхраняват уникален идентификатор (ID), три имена, факултетен номер (който трябва да бъде уникален в рамките на системата) и специалност.
- **Преподаватели:** Академичният състав, дефиниран чрез име и научна титла/длъжност (напр. Доцент, Професор).
- **Учебни дисциплини (Курсове):** Предметите, които се изучават. Всеки курс е логически обвързан с един водещ преподавател.
- **Оценяване:** Процесът на записване на резултат от изпит. Това представлява връзка от тип „Много-към-Много“ между Студент и Дисциплина, която се характеризира с атрибута „Оценка“.

Функционални изисквания:

1. Възможност за динамично добавяне на нови записи във всички номенклатури.
2. Защита от дублиране на данни (напр. двама студенти с един и същ факултетен номер).
3. Генериране на справка за индивидуален успех на студент.

4.4. Методология на работа

За реализацията на проекта е възприет итеративен подход за разработка на софтуер (Iterative Model), преминаващ през четири основни етапа:

Етап 1: Проектиране на данните (Database Design)

Първоначално бе дефинирана схемата на базата данни. Бяха идентифицирани таблиците и връзките между тях. Този етап е критичен, тъй като лошата структура на базата данни би довела до проблеми при писането на кода.

Етап 2: Разработка на „Backend“ логика

Създаден е специализиран клас DB (Database Handler). Този клас прилага шаблона за дизайн DAO (Data Access Object). Той капсулира цялата комуникация с базата данни. Графичният интерфейс не пише директни SQL заявки, а извиква методи на този клас (напр. db.add_student()). Това осигурява модулност и лесна поддръжка на кода.

Етап 3: Изграждане на „Frontend“ (GUI)

След като логиката за данните бе готова, бе разработен класът UniversityApp. Той е отговорен за визуализацията. Използван е компонентът ttk.Notebook за създаване на навигация чрез табове. На този етап бяха свързани събитията от бутоните (Events) с методите от Backend логиката.

Етап 4: Тестване и Дебъгване (Testing & Debugging)

Последният етап включваше въвеждане на тестови данни (както коректни, така и некоректни) за проверка на устойчивостта на системата. Бяха отстранени грешки, свързани с типовете данни (напр. опит за въвеждане на текст в поле за оценка) и бяха добавени съобщения за потвърждение при изтриване на данни.

5. РЕЗУЛТАТИ И ОБСЪЖДАНЕ

В този раздел се представя детайлно техническата реализация на информационната система, като се разглеждат архитектурата на базата данни, програмният код и потребителският интерфейс.

5.1. Структура на базата данни

Проектираната база данни university_pro.db е реализирана съгласно принципите на релационния модел. За да се избегне излишното дублиране на данни и да се осигури логическа свързаност, структурата е нормализирана до **Трета нормална форма (3NF)**.

Базата се състои от четири основни таблици. По-долу е представено детайлно описание на всяка от тях и SQL заявките (DDL), използвани за създаването им.

5.1.1. Таблица students (Студенти)

Това е основната номенклатура в системата. Тя съхранява идентификационните данни за всеки обучаем.

- **id (INTEGER PRIMARY KEY AUTOINCREMENT):** Уникален системен идентификатор. Генерира се автоматично от базата данни.
- **name (TEXT NOT NULL):** Пълно име на студента. Полето е задължително.
- **fn (TEXT UNIQUE NOT NULL):** Факултетен номер. Дефиниран е като уникален ключ (UNIQUE), което предотвратява възможността двама студенти да бъдат записани с еднакъв номер.
- **major (TEXT):** Специалност на студента.
-

```
CREATE TABLE IF NOT EXISTS students (
id INTEGER PRIMARY KEY AUTOINCREMENT,
name TEXT NOT NULL,
fn TEXT UNIQUE NOT NULL,
major TEXT NOT NULL
);
```

5.1.2. Таблица professors (Преподаватели)

Тази таблица служи за управление на преподавателския състав. Отделянето на преподавателите в самостоятелна таблица позволява един преподавател да бъде свързван с множество курсове, без да се налага многократно въвеждане на името му.

- **title (TEXT):** Академична длъжност (напр. „Доцент“, „Гл. Асистент“).
-

```
CREATE TABLE IF NOT EXISTS professors (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   name TEXT NOT NULL,
   title TEXT
);
```

5.1.3. Таблица courses (Учебни дисциплини)

Тази таблица съхранява списъка с изучаваните предмети. Тук се реализира връзка тип **„Едно-към-Много“ (One-to-Many)** с таблицата на преподавателите.

- **professor_id (INTEGER):** Външен ключ (FOREIGN KEY), който сочи към id в таблица professors.
- **Ограничение ON DELETE SET NULL:** Ако преподавател бъде изтрит от системата, курсът не се изтрива, а полето за преподавател автоматично става празно (NULL). Това е защитен механизъм за запазване на учебната програма.
-

```
CREATE TABLE IF NOT EXISTS courses (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   name TEXT NOT NULL,
   professor_id INTEGER,
   FOREIGN KEY (professor_id) REFERENCES professors (id) ON DELETE SET NULL
);
```

5.1.4. Таблица grades (Оценки и Протоколи)

Това е асоциативна (свързваща) таблица, която реализира връзката **„Много-към-Много“ (Many-to-Many)** между студенти и курсове.

- **student_id:** Връзка към студент.
- **course_id:** Връзка към дисциплина.
- **grade (REAL):** Числова стойност на оценката.
- **Ограничение ON DELETE CASCADE:** Това е критично важна настройка за цялостта на данните. Тя гарантира, че ако запис на студент бъде изтрит от таблица students, базата данни автоматично ще изтрие всички свързани с него оценки в таблица grades. Така се предотвратява появата на т.нар. „сираци“ (orphaned records) – оценки, които не принадлежат на никого.
-

```
CREATE TABLE IF NOT EXISTS grades (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   student_id INTEGER,
   course_id INTEGER,
   grade REAL,
   FOREIGN KEY (student_id) REFERENCES students (id) ON DELETE CASCADE,
   FOREIGN KEY (course_id) REFERENCES courses (id) ON DELETE CASCADE
);
```

5.2. Структура на програмата

Приложното решение е разработено чрез **Обектно-Ориентирано Програмиране (ООР)**. Логиката е разделена на два основни класа, което спазва принципа за „Разделяне на отговорностите“ (Separation of Concerns).

5.2.1. Клас DB (Backend Logic)

Този клас изпълнява ролята на слой за достъп до данни (Data Access Layer). Той енкапсулира (скрива) сложността на SQL заявките от останалата част на програмата.

Основни характеристики:

- **Инициализация:** В конструктора __init__ се осъществява връзката с базата данни и се активира поддръжката на Foreign Keys (PRAGMA foreign_keys = 1), която по подразбиране е изключена в SQLite.
- **Сигурност (Parameterized Queries):** Всички методи за запис на данни използват параметризирани заявки (чрез символа ?). Вместо директно вмъкване на текст в SQL низа (string concatenation), данните се подават като отделен аргумент. Това напълно елиминира риска от **SQL Injection** атаки, при които злонамерен потребител би могъл да манипулира базата чрез входните полета.

Пример за защитен метод:
Python

```
def add_student(self, name, fn, major):
    # Данните се подават като tuple във втория аргумент, а не се залепят за стринга
    self.cur.execute("INSERT INTO students ... VALUES (?, ?, ?)", (name, fn, major))
```

5.2.2. Клас UniversityApp (Frontend Logic)

Този клас наследява функционалностите на Tkinter и управлява целия потребителски интерфейс.

- **Управление на състоянието:** Класът поддържа вътрешни речници (dictionaries) – self.map_students, self.map_courses. Те служат за mapping (свързване) между имената, които потребителят вижда на екрана (напр. "Иван Иванов"), и техните ID номера в базата данни (напр. 42).
- **Event Handling:** Реализиран е механизъм за обновяване на данните при смяна на табовете (<<NotebookTabChanged>>). Това гарантира, че ако потребителят добави нов студент в първия таб, името му веднага ще се появи в падащото меню за оценки в четвъртия таб.

5.3. Графичен интерфейс (GUI)

Интерфейсът е реализиран чрез библиотеката **ttkbootstrap**, която прилага темата "Superhero". Това осигурява тъмен режим (Dark Mode), висока контрастност и модерни заоблени форми на бутоните, което значително подобрява потребителското изживяване (UX) в сравнение със стандартния Tkinter.

Приложението е организирано в навигационна структура с 4 таба:

1. **Таб "Студенти":**
   - *Лява част / Форма:* Полета за въвеждане на Име, Фак. номер и Специалност.
   - *Дясна част / Таблица:* Използван е компонент Treeview за визуализация на списъка със студенти.
   - *(Тук постави Скрийншот 1: Екран Студенти)*
2. **Таб "Преподаватели":**
   - Позволява въвеждане на академичен състав.
   - *(Тук постави Скрийншот 2: Екран Преподаватели)*
3. **Таб "Дисциплини":**
   - Ключов елемент тук е падащото меню (Combobox), което динамично зарежда списъка с налични преподаватели от базата данни. Това предотвратява грешки при въвеждане на имена.
   - *(Тук постави Скрийншот 3: Екран Дисциплини)*
4. **Таб "Оценки (Протокол)":**
   - Най-комплексният екран. Съдържа два свързани списъка (Студент и Дисциплина) и поле за оценка.
   - Таблицата визуализира резултатите чрез **SQL JOIN**, показвайки четими имена, а не цифрови идентификатори.
   - *(Тук постави Скрийншот 4: Екран Оценки)*

5.4. Функционални резултати и тестови сценарии

По време на тестовата фаза системата демонстрира стабилна работа при изпълнение на основните бизнес процеси.

Сценарий 1: Валидация на данни

- *Действие:* Опит за регистрация на студент с Факултетен номер, който вече съществува в базата.
- *Резултат:* Системата прихваща грешката sqlite3.IntegrityError и извежда диалогов прозорец със съобщение: "Грешка: Дублиран Фак. номер!". Записът не се създава.

Сценарий 2: Изчисляване на Среден успех (GPA)

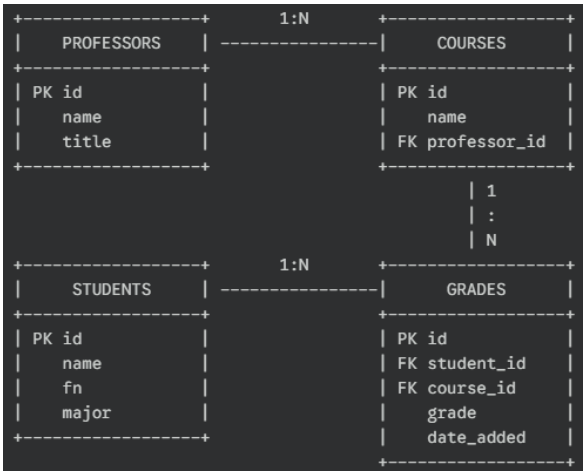
- *Действие:* Избор на студент от таблицата и натискане на бутон "Справка Успех".

- *Логика:* Системата изпълнява агрегираща заявка: SELECT AVG(grade) FROM grades WHERE student_id = ?.
- *Резултат:* Резултатът се закръгля до втория знак и се показва в информационен прозорец.

5.5. Диаграми и таблици

А) ER Диаграма (Схема на базата данни)

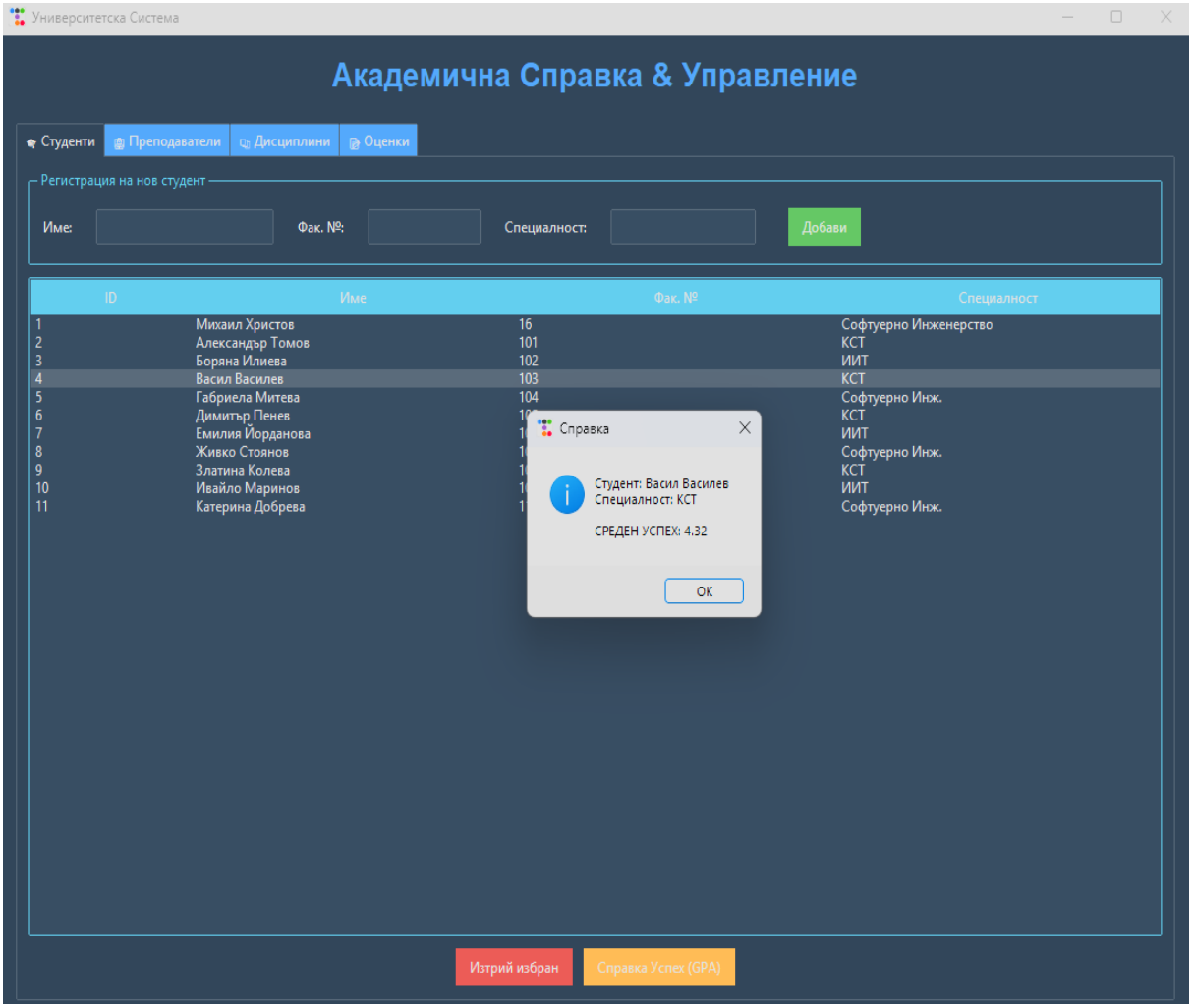
Фигурата по-долу илюстрира връзките между същностите в системата. Ясно се вижда централната роля на таблицата grades, която свързва студентите и учебните дисциплини.



Б) Таблица на функционалностите

Функционалност	**Реализация (Backend)**	**Реализация (GUI)**
Добавяне на студент	INSERT INTO students	ttk.Entry + Бутон
Свързване Курс-Преподавател	FOREIGN KEY	ttk.Combobox
Проверка за валидна оценка	Python if 2 <= grade <= 6	Валидация при клик
Изтриване на данни	DELETE ... CASCADE	Диалог за потвърждение
Изчисляване на GPA	SQL AVG() функция	Pop-up прозорец

Фигура 1: Модул „Студенти“

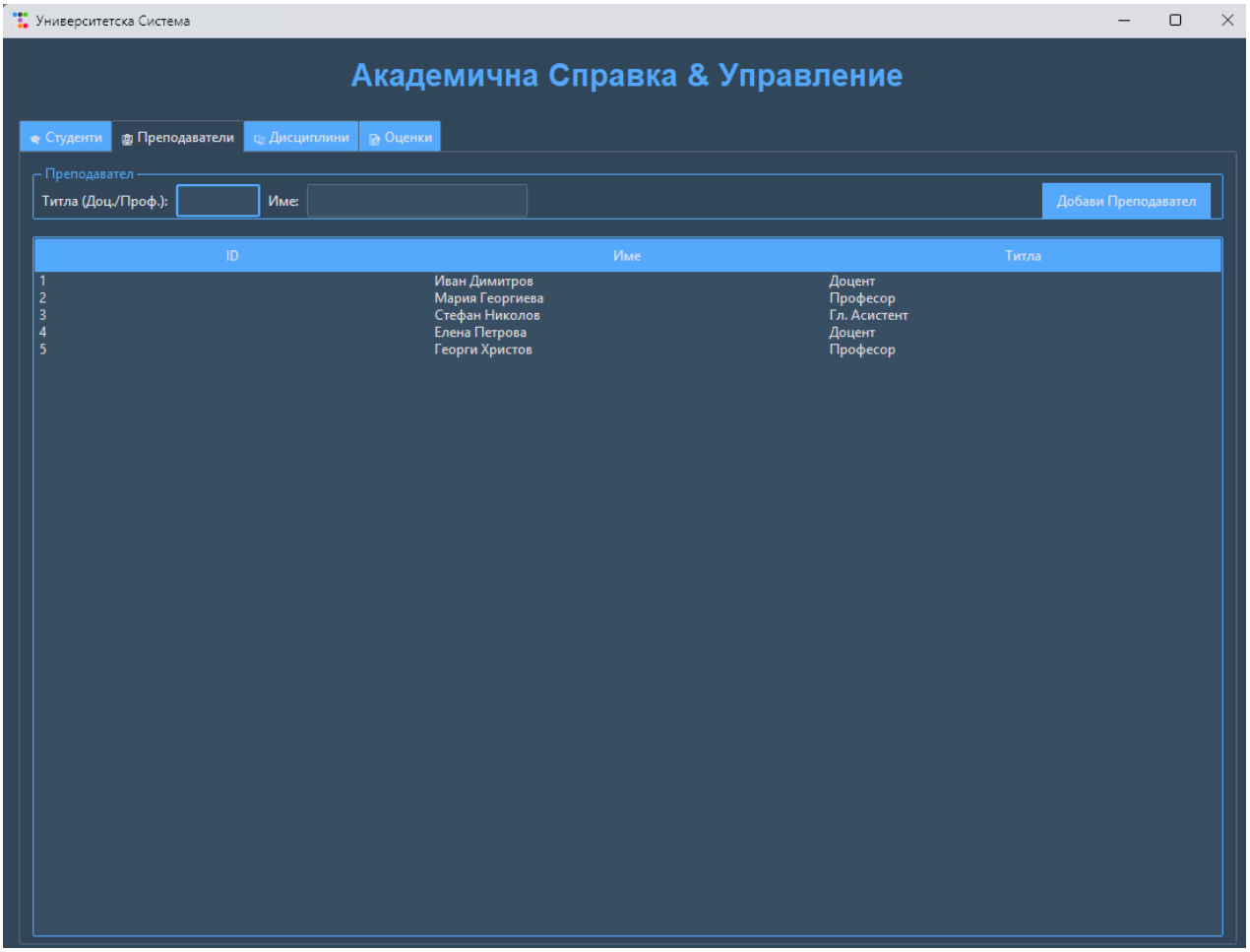


Фигура 1. Основен екран за управление на студентския състав и регистрация.

Описание:

На фигурата е представен основният работен екран на модул „Студенти“. В горната част е разположена формата за регистрация, съдържаща полета за трите ключови атрибута: „Име“, „Факултетен номер“ и „Специалност“. Използван е компонентът Treeview (таблицата в центъра) за визуализация на текущите записи в базата данни. Вижда се, че интерфейсът прилага темата "Superhero", осигурявайки висок контраст и четимост. В долната част са разположени контролните бутони за изтриване на запис и за генериране на справка за успех, което осигурява бърз достъп до най-често използваните функции от администратора.

Фигура 2: Модул „Преподаватели“

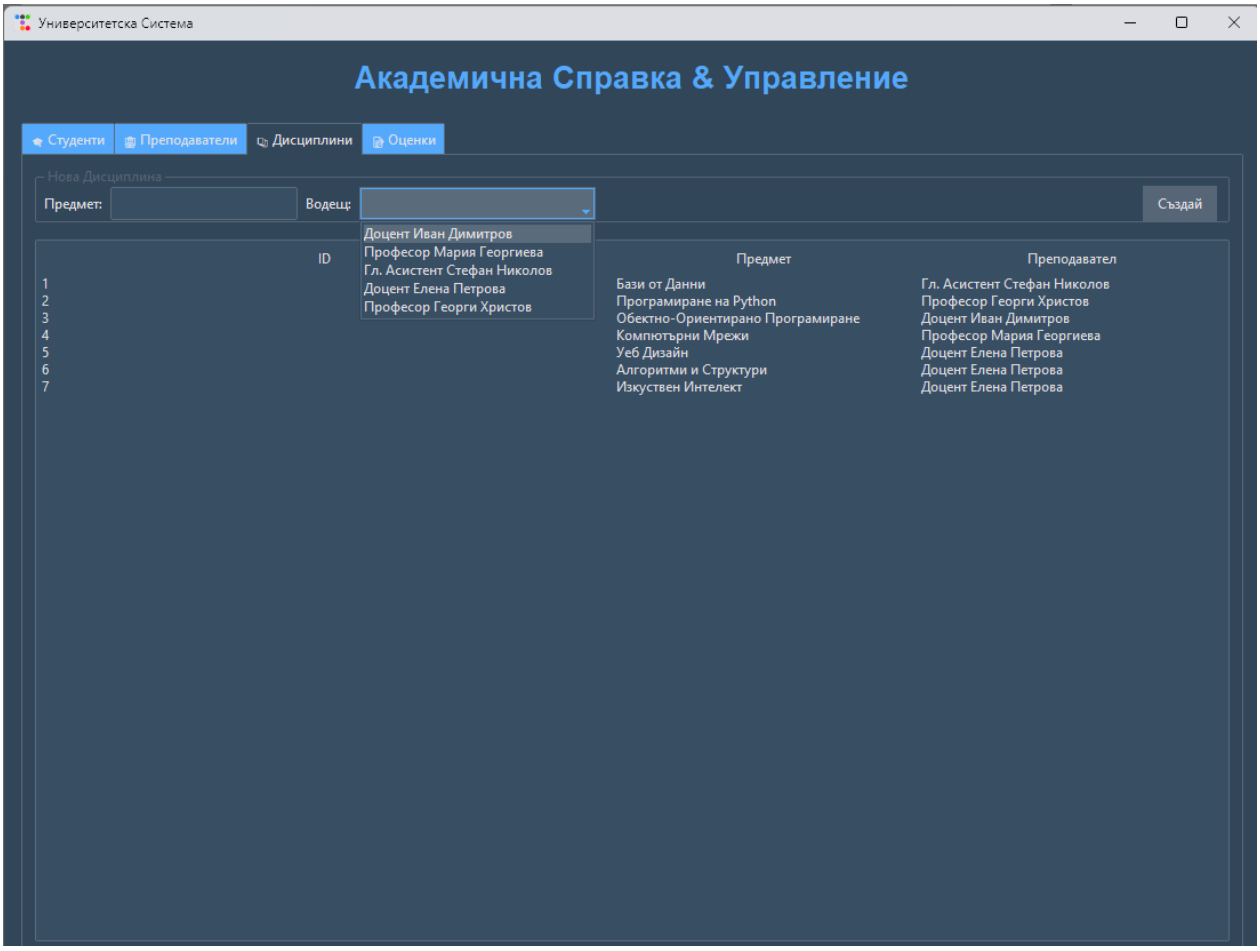


Фигура 2. Интерфейс за администриране на академичния състав.

Описание:

Този екран демонстрира попълването на номенклатурата с преподаватели. За разлика от студентите, тук се изисква въвеждане на академична длъжност (Титла). Данните от тази таблица са критични за целостта на системата, тъй като те ще бъдат използвани като външен ключ (Foreign Key) при създаването на курсове. Визуализацията потвърждава успешното записване на кирилица и коректното извличане на данните от таблицата professors чрез SELECT заявка.

Фигура 3: Създаване на дисциплини и връзки

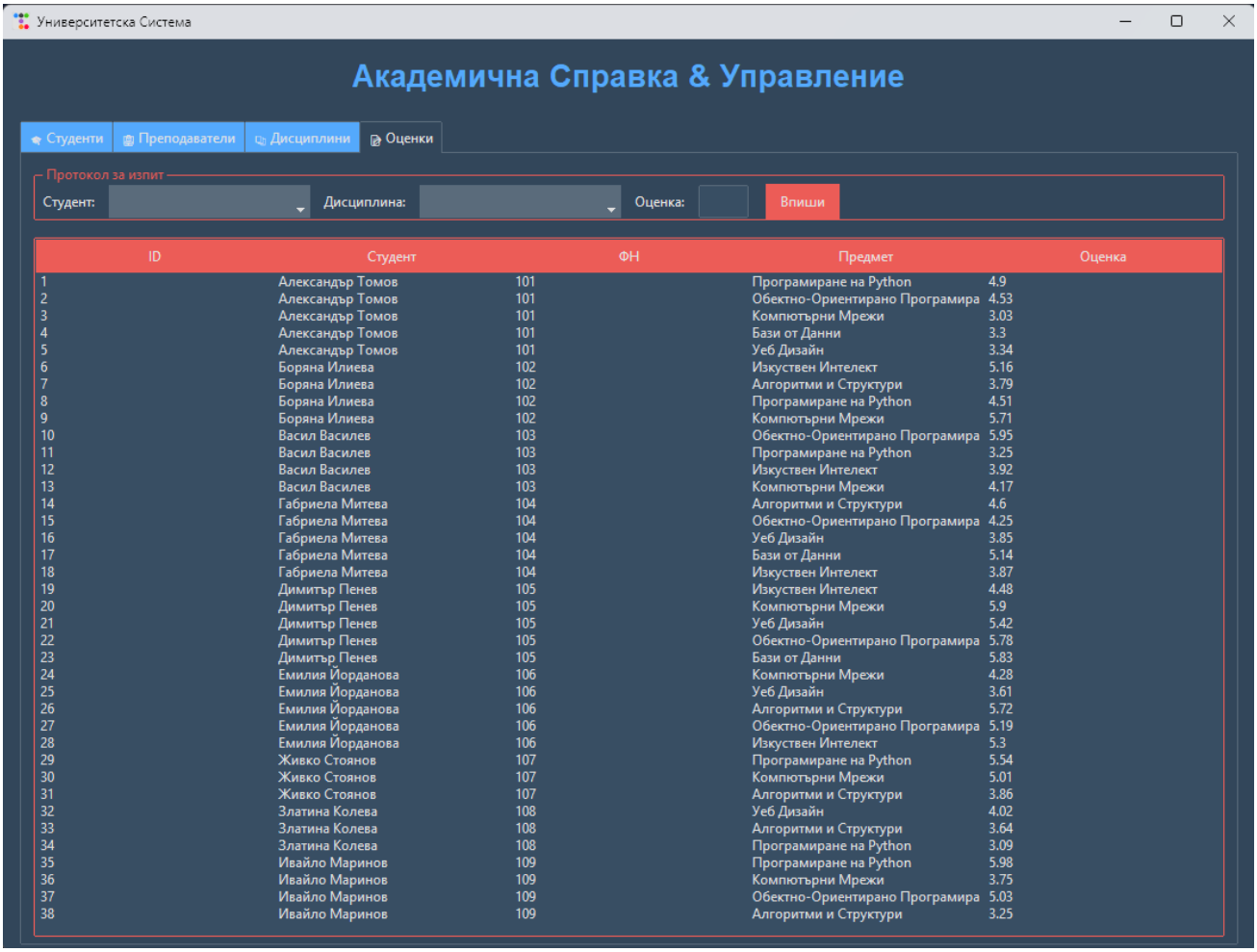


Фигура 3. Създаване на учебна дисциплина с назначаване на водещ преподавател.

Описание:

Фигурата илюстрира реализацията на релационната връзка „Едно-към-Много“ (One-to-Many). При създаването на нова дисциплина, полето „Водещ“ е реализирано чрез падащ списък (Combobox). Този списък се попълва динамично при стартиране на програмата или смяна на таба, като извлича наличните преподаватели от базата данни. Това техническо решение предотвратява възможността за потребителска грешка – администраторът не може да въведе несъществуващ преподавател, което гарантира референтния интегритет на базата данни.

Фигура 4: Изпитен протокол (Оценки)

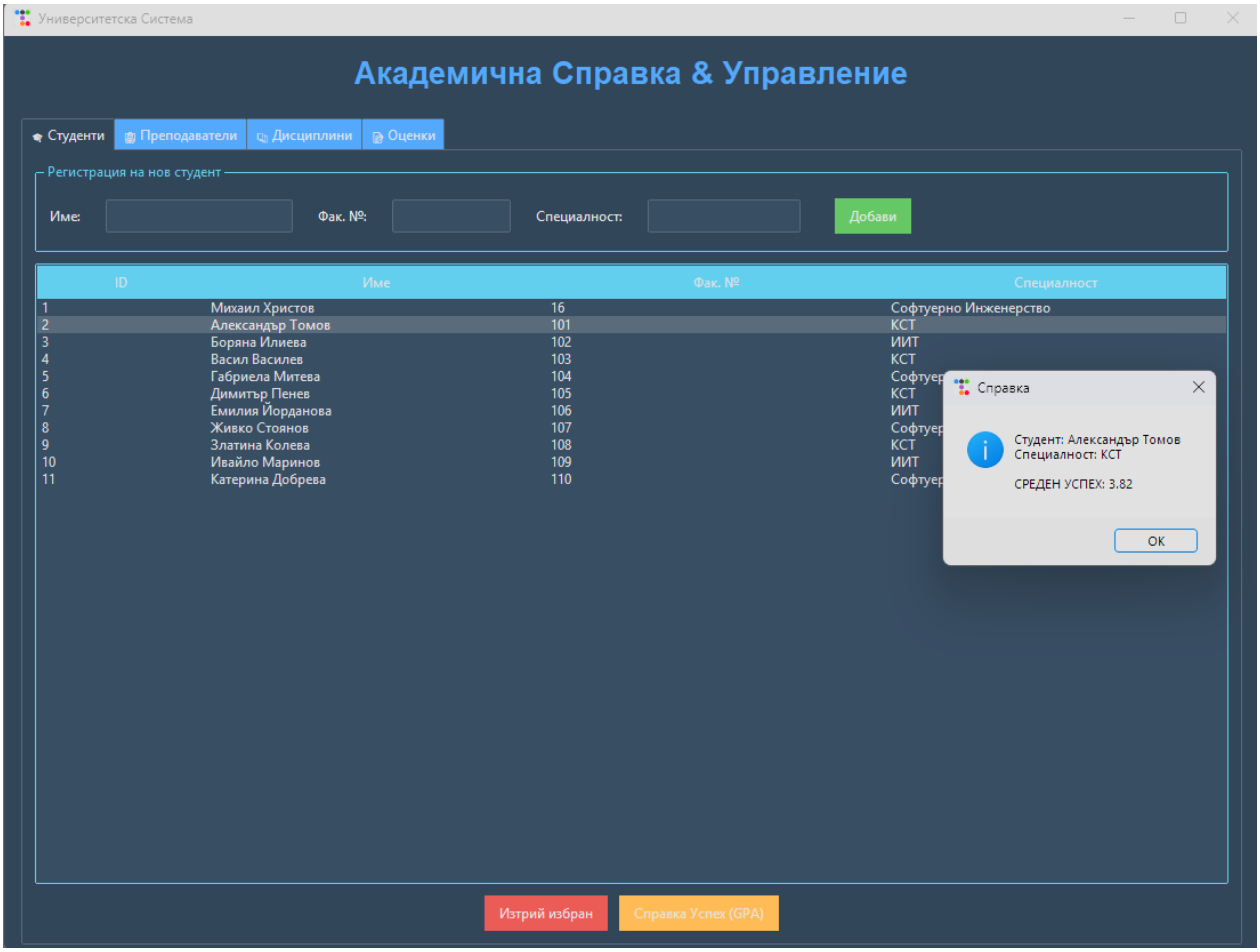


Фигура 4. Електронен изпитен протокол и визуализация на оценките.

Описание:

Това е най-сложният модул на системата, реализиращ връзката „Много-към-Много“. На екрана се вижда процесът на оценяване: избор на студент и дисциплина от списъци и въвеждане на цифрова оценка. Важно е да се отбележи, че въпреки че в базата данни (таблица grades) се пазят само ID номерата (напр. student_id=5, course_id=2), графичният интерфейс извършва SQL заявка с JOIN, за да покаже на потребителя четимите имена на студента и предмета. Това прави системата удобна за работа, скривайки техническата сложност от крайния потребител.

Фигура 5: Генериране на справка (GPA)

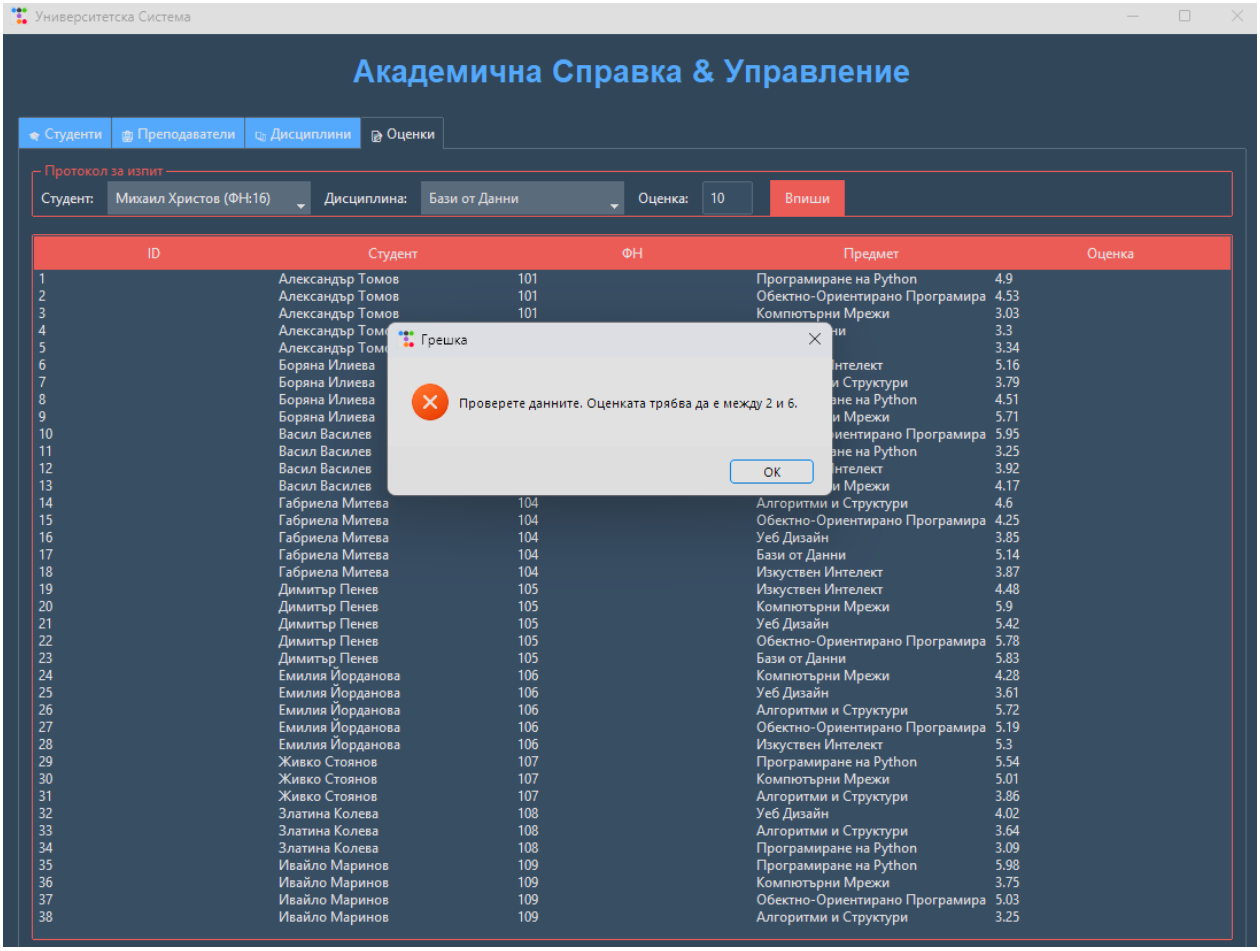


Фигура 5. Автоматизирано генериране на справка за среден успех.

Описание:

Фигурата показва резултата от изпълнението на бизнес логиката на приложението. При избор на конкретен студент и натискане на функционалния бутон, системата извършва агрегираща заявка към базата данни (SELECT AVG(grade)...), изчислява средната аритметична стойност на всичките му оценки и я визуализира в модален диалогов прозорец. Резултатът е закръглен до втори знак след десетичната запетая, съгласно академичните стандарти.

Фигура 6: Валидация на данни (Грешка)



Фигура 6. Механизъм за валидация и обработка на грешки.

Описание:

За да се гарантира качеството на данните, системата разполага с вградени валидационни механизми. На фигурата е показан реакцията на софтуера при опит за въвеждане на невалидни данни (в случая – оценка извън допустимия диапазон 2.00–6.00). Приложението прихваща некоректния вход, прекратява записа в базата данни и информира потребителя чрез съобщение за грешка. Това демонстрира устойчивостта на системата срещу неправилна употреба.

6. ИЗВОДИ И ПРЕПОРЪКИ

В резултат на проведения анализ, проектиране и реализация на софтуерната система, могат да бъдат направени следните основни изводи относно избрания технологичен подход и постигнатите резултати.

6.1. Изводи от разработката

1. **Ефективност на релационния модел:** Използването на релационна база данни (SQLite) и стриктното спазване на правилата за нормализация (3NF) се доказа като критичен фактор за стабилността на приложението. Чрез дефинирането на външни ключове (Foreign Keys) и правила за интегритет (ON DELETE CASCADE) бе постигната висока надеждност на данните, елиминирайки възможността за логически грешки и дублиране на информация.
2. **Баланс между функционалност и сложност:** Изборът на Python като език за разработка позволи бързо прототипиране и реализация (Rapid Application Development). Модулната архитектура на кода, разделяща логиката за базата данни от потребителския интерфейс, улесни процеса на тестване и поддръжка.
3. **Потребителско преживяване (UX):** Интеграцията на библиотеката ttkbootstrap демонстрира, че десктоп приложенията, писани на Python, могат да имат модерен и професионален вид. Използването на тъмна тема ("Superhero") и интуитивни контроли (падащи менюта, таблици) значително намалява когнитивното натоварване на потребителя и повишава ефективността на работа.
4. **Валидация на входните данни:** Вграждането на проверки на ниво приложение (Application Layer) – като ограничаване на оценките в диапазона 2-6 и проверка за уникален факултетен номер – се оказа ефективна първа линия на защита срещу човешки грешки.

6.2. Препоръки за бъдещо развитие

Въпреки че текущата версия на системата покрива базовите изисквания за управление на учебен процес, съществуват възможности за нейното надграждане и оптимизация в следните направления:

- Модул за автентификация (Login System):

  Към момента системата е достъпна за всеки, който има достъп до компютъра. Препоръчва се добавяне на входен екран с потребителско име и парола. Паролите трябва да се съхраняват в базата данни не като чист текст, а като хеш-стойности (използвайки алгоритми като SHA-256 или библиотеката bcrypt) за по-висока сигурност. Също така може да се реализират роли (напр. "Администратор" с пълни права и "Преподавател" с права само за писане на оценки).

- Експорт и отчетност:

За да бъде системата пълноценна в реална среда, е необходимо да се реализира функционалност за експорт на справките. Използването на библиотеки като pandas (за Excel) и reportlab (за PDF) би позволило генерирането на официални академични справки и уверения, готови за печат.

- Разширени аналитики:

  Добавяне на модул за визуализация на данните (Dashboard). Чрез библиотека като matplotlib могат да се генерират графики за средния успех по години, разпределение на оценките по дисциплини и сравнителен анализ между различни специалности.

- Миграция към Клиент-Сървър архитектура:

  При необходимост от едновременна работа на много служители, базата данни SQLite може да бъде заменена с PostgreSQL или MySQL, без това да изисква промяна в графичния интерфейс, благодарение на модулната структура на класа DB.

7. ЛИТЕРАТУРА И ИЗПОЛЗВАНИ ИЗТОЧНИЦИ

За теоретичната обосновка и практическата реализация на курсовия проект са използвани следните литературни и електронни източници:

1. Python 3 Documentation. Python Software Foundation. Достъпно на: docs.python.org/3/
2. SQLite Syntax Specification & Documentation. Достъпно на: sqlite.org/docs.html
3. Ttkbootstrap Documentation: A supercharged theme extension for tkinter. Достъпно на: ttkbootstrap.readthedocs.io
   (Използвано за стилизиране на графичния интерфейс).
4. Клипове и уроци от codecademy и Youtube.
5. Помощ от DevOps приятел и Gemini.

8. ЗАКЛЮЧЕНИЕ

Разработката на курсовия проект на тема „Академична Информационна Система“ премина през всички етапи на жизнения цикъл на софтуера – от анализ на изискванията и проектиране на архитектурата, през кодиране и тестване, до финализиране на документацията.

Поставените цели и задачи бяха изпълнени в пълен обем:

- Проектирана е устойчива релационна база данни в 3NF.
- Реализиран е интуитивен графичен интерфейс, улесняващ работата на администратора.
- Имплементирани са всички необходими CRUD операции за управление на студенти, преподаватели, дисциплини и оценки.
- Осигурена е защита на целостта на данните чрез каскадно изтриване и валидация.

Проектът демонстрира успешното прилагане на теоретичните знания в практиката. Създаденото приложение е стабилно, лесно за разширение и може да послужи както за учебни цели, така и като основа за реална система в малки образователни звена. Работата по проекта затвърди уменията за работа с обектно-ориентирано програмиране, SQL заявки и проектиране на потребителски интерфейси, които са фундаментални за професионалното развитие в сферата на информационните технологии.