

АВТОНОМНАЯ НЕКОММЕРЧЕСКАЯ ОРГАНИЗАЦИЯ  
ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ  
МОСКОВСКИЙ МЕЖДУНАРОДНЫЙ КОЛЛЕДЖ ЦИФРОВЫХ ТЕХНОЛОГИЙ  
«АКАДЕМИЯ ТОП»

**ПРОЕКТ**

**по дисциплине «Технология доступа к базам данных ADO.NET»**

**«Разработка системы онлайн-обучения»**

Выполнили:

студенты группы: 9/3-РПО-23/2

Карпухова А.А, Ерофеев К.П, Тулуш И.

Преподаватель:

Рослова О.А

г. Санкт-Петербург

2025

## **Введение**

**Цель:** Разработать стабильное, безопасное и производительное консольное приложение, связанное с онлайн-обучением. Основная техническая особенность - эффективное использование [ado.net](https://docs.microsoft.com/ru-ru/ado/ado-overview) для взаимодействия с базой данных, обеспечивая быстрое выполнение запросов, целостность данных и безопасность от внешних угроз (sql-инъекций). В качестве ado будет использоваться LINQ .

### **Задачи проекта:**

1. Разработать базу данных, заполнить таблицы и нормализовать ее.
2. Организовать работу с базой данных:
  - добавление/удаление пользователей.
  - редактирование данных таблиц.
  - просмотр данных с возможностью сортировки.
  - регистрация пользователя.
  - хэширование и восстановление пароля по почте(без отправки сообщения на почту!)
3. Создать интуитивно - понятный консольный интерфейс.
4. Протестировать консольное приложение и базу данных.
5. Написать вывод, создать презентацию и защитить проект.

**Язык программирования:** C#

**СУБД:** MS SQL Server + SSMS

**Программа для реализации:** MS Visual Studio

**Объекты базы данных(сущности):** users, course, level access, wallet

## LINQ to Entities

LINQ to Entities — это технология в C#, которая позволяет писать запросы к базе данных, используя знакомый синтаксис LINQ и объекты вашей предметной области (Entity Classes), вместо прямого написания SQL-запросов. Проще говоря, это "переводчик" между вашими LINQ-запросами на C# и SQL-запросами, которые понимает база данных (например, SQL Server, PostgreSQL, MySQL).

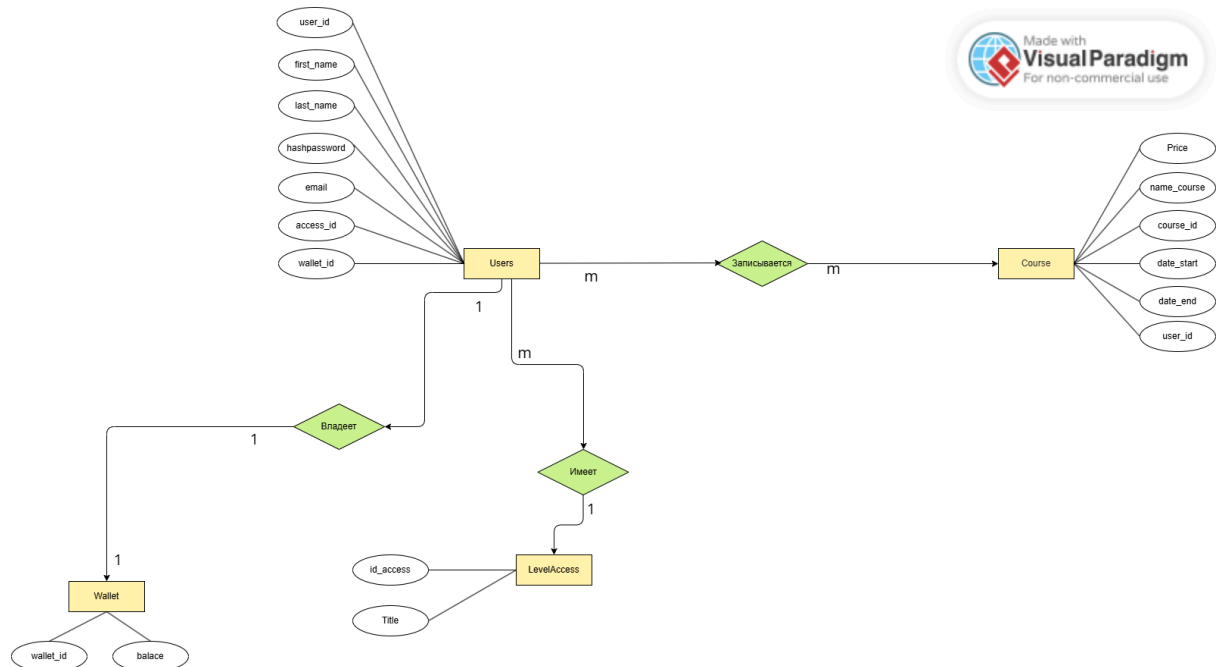
Ключевые преимущества:

1. **Безопасность типов (Type Safety):** Компилятор C# проверяет ваши запросы на корректность. Если вы опечатались в имени свойства (City вместо City), вы получите ошибку на этапе компиляции, а не во время выполнения.
2. **Продуктивность разработчика:** Не нужно писать и хранить громоздкие SQL-строки. Используется IntelliSense, рефакторинг и все возможности C#.
3. **Абстракция от БД:** Вы пишете код на C#, а EF и LINQ to Entities заботятся о генерации специфичного для вашей СУБД SQL. Теоретически, вы можете сменить базу данных с минимальными изменениями в коде.
4. **Читаемость и поддерживаемость:** Код запросов интегрирован в C# и следует той же логике и структуре, что и остальная часть приложения.

LINQ to Entities — это мощный и удобный мост между объектно-ориентированным миром C# и реляционным миром баз данных. Он является неотъемлемой частью Entity Framework и позволяет разработчикам работать с данными на высоком уровне абстракции, не теряя при этом в производительности.

# Проектирование базы данных

## Er-модель системы онлайн-обучения



## Основные сущности и их назначение

### 1. Users (Клиент/Студент/Преподаватель)

Атрибуты:

- client\_id - уникальный идентификатор клиента
- first\_name, last\_name - имя и фамилия
- email - почта
- password - хэш пароль для аутентификации
- level\_access\_id - идентификатор роли
- wallet\_id - идентификатор кошелька(Id = Null, если преподавателю не нужен кошелек)

### 2. LevelAccess (Уровень доступа)

Атрибуты:

- id\_access - идентификатор уровня доступа
- Role - роль пользователя

### 3. Course (Курс)

Атрибуты:

- course\_id - уникальный идентификатор курса
- name\_course - название курса
- date\_start, date\_end - даты начала и окончания курса
- user\_id - идентификатор преподавателя
- Price - цена на курс (цена может быть ноль)

### 4. Wallet (Кошелек)

Атрибуты:

- wallet\_id - уникальный идентификатор кошелька
- balance - баланс средств

### Связи между сущностями

1. Users  $\longleftrightarrow$  Course: Связь многие-ко-многим. Так как один студент/преподаватель может записываться на множество курсов и один курс может иметь множество студентов/преподавателей. (Будет реализовано через составной ключ)

2. Users  $\longleftrightarrow$  Wallet: Связь один-к-одному. Так как каждый студент/преподаватель имеет только один кошелек и каждый кошелек принадлежит одному студенту. Если преподаватель не нужен кошелек то id = null.

3. Users  $\longleftrightarrow$  LevelAccess: Связь многие-к-одному. Так как многие студенты/преподаватели имеют только одну роль Student/Teacher.

## Нормализация БД

### Не нормализованная таблица

**Таблица: Users**

user_id	first_name	last_name	email	password	level_access_id	wallet_id
1	Иван	Петров	<a href="mailto:ivan@mail.ru">ivan@mail.ru</a>	pass123	1	1
2	Мария	Иванова	<a href="mailto:maria@yandex.ru">maria@yandex.ru</a>	pass456	1	2
3	Алексей	Смирнов	<a href="mailto:alex@gmail.com">alex@gmail.com</a>	pass789	2	3

**Таблица: Wallet**

wallet_id	balance
1	1000
2	1500
3	2000

**Таблица: LevelAccess**

level_id	Role
1	Student
2	Admin
3	Teacher
4	banned

**Таблица: Course**

course_id	course_name	date_start	date_end	user_id	Price
1	Python Basics	2024-01-15	2024-03-15	1	10000
2	Web Development	2024-02-01	2024-04-01	2	15000
3	Data Science	2024-01-10	2024-03-10	1	20000

## 1NF

### Таблица: Users

user_id	first_name	last_name	email	password	level_access_id	wallet_id
1	Иван	Петров	<a href="mailto:ivan@mail.ru">ivan@mail.ru</a>	pass123	1	1
2	Мария	Иванова	<a href="mailto:maria@yandex.ru">maria@yandex.ru</a>	pass456	1	2
3	Алексей	Смирнов	<a href="mailto:alex@gmail.com">alex@gmail.com</a>	pass789	2	3

### Таблица: Wallet

wallet_id	balance
1	1000
2	1500
3	2000

### Таблица: LevelAccess

level_id	Role
1	Student
2	Admin
3	Teacher
4	banned

### Таблица: Course

course_id	course_name	date_start	date_end	user_id	Price
1	Python Basics	2024-01-15	2024-03-15	1	10000
2	Web Development	2024-02-01	2024-04-01	2	15000
3	Data Science	2024-01-10	2024-03-10	1	20000

Анализ исходных данных:

- Все атрибуты атомарны
- Нет повторяющихся групп данных
- Результат: Все таблицы соответствуют 1nf.

## 2NF

### Таблица: Users

user_id	first_name	last_name	email	password	level_access_id	wallet_id
1	Иван	Петров	<a href="mailto:ivan@mail.ru">ivan@mail.ru</a>	pass123	1	1
2	Мария	Иванова	<a href="mailto:maria@yandex.ru">maria@yandex.ru</a>	pass456	1	2
3	Алексей	Смирнов	<a href="mailto:alex@gmail.com">alex@gmail.com</a>	pass789	2	3

### Таблица: Wallet

wallet_id	balance
1	1000
2	1500
3	2000

### Таблица: LevelAccess

level_id	Role
1	Student
2	Admin
3	Teacher
4	banned

### Таблица: Course

course_id	course_name	date_start	date_end	user_id	Price
1	Python Basics	2024-01-15	2024-03-15	1	10000
2	Web Development	2024-02-01	2024-04-01	2	15000
3	Data Science	2024-01-10	2024-03-10	1	20000

### Анализ исходных данных

- Все таблицы имеют простые первичные ключи
- Все неключевые атрибуты полностью зависят от первичных ключей

Результат: Все таблицы соответствуют 2nf.



### 3NF

**Таблица: LevelAccess**

level_id (PK)	Title
1	Student
2	admin
3	Teacher
4	banned

**Таблица: Users**

user_id (PK)	first_name	last_name	email	password	level_access_id	wallet_id
1	Main	Первое	http@mailuser	pass123	1	1
2	Марки	Иванова	<a href="mailto:mail@gradator.ru">mail@gradator.ru</a>	pass456	1	2
3	Алексей	Смирнов	<a href="mailto:abs@email.com">abs@email.com</a>	pass789	2	3

**Таблица: Wallet**

wallet_id (PK)	balance
1	1000
2	1500
3	2000

**Таблица: Course**

course_id (PK)	course_name	date_start	date_end	user_id	Price
1	Python Basics	2024-01-15	2024-03-15	1	10000
2	Web Development	2024-02-01	2024-04-01	2	10000
3	Data Science	2024-01-10	2024-03-10	1	20000

**Таблица: ClientCourse**

ClientCourse_id (PK)	user_id	course_id
1	1	1
2	2	2
3	3	3

Анализ исходных данных:

- Отсутствует таблица связи многие-ко-многим между Users и Course

Исправление:

- Создана таблица UserCourses для связи клиентов и курсов

Описание типов данные и ограничений:

**Users:**

- id\_user (int PK)
- first\_name (varchar(255) not NULL)
- last\_name (varchar(255) not NULL)
- password (varchar(255) not NULL)
- level\_access\_id (int FK not null)
- email (varchar(255) not NULL unique)
- wallet\_id (int FK)

**LevelAccess:**

- id\_access (int PK)
- Title (varchar(255) not NULL unique)

**Course:**

- id\_course (int PK)
- name\_course (varchar(255) not NULL)
- date\_start (DateTime not null)
- date\_end (DateTime not null)
- user\_id (int FK not null)
- Price (int not NULL DEFOULT = 0)
- user\_id

**Wallet:**

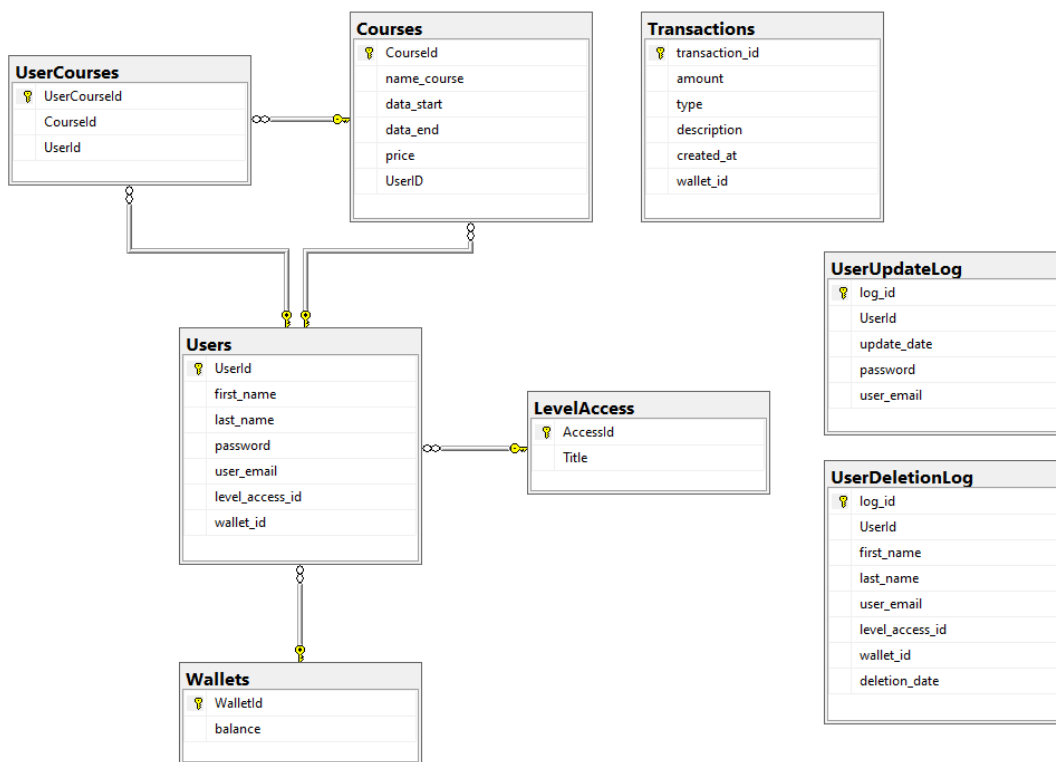
- id\_wallet (int PK)

- balance (float Defount = 0)

## UserCourses

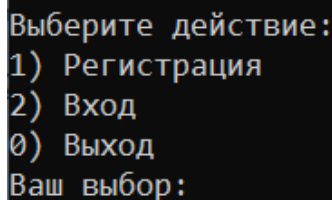
- id (int PK)
- User\_Id (int FK not null)
- Course\_Id (int FK not null)

## Таблица реализованная в СУБД



## Интерфейс программы

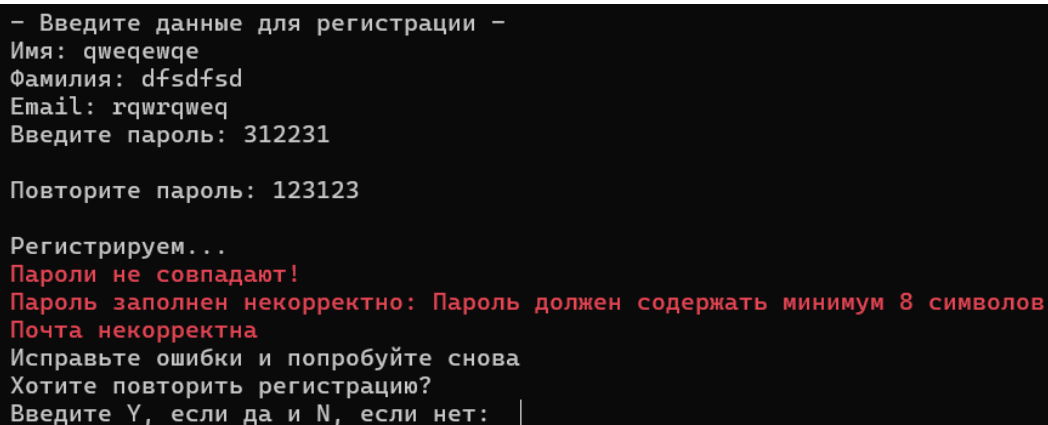
1. После запуска программы, выводится начальный интерфейс (Рисунок 1). В котором мы можем выбрать “Регистрация”, “Вход” и “Выход”.



```
Выберите действие:  
1) Регистрация  
2) Вход  
0) Выход  
Ваш выбор:
```

Рисунок 1. Начальный интерфейс

2. Регистрация аккаунта с валидацией данных (Рисунок 2, 3).



```
- Введите данные для регистрации -  
Имя: qweqewqe  
Фамилия: dfsdfsd  
Email: rqwrqweq  
Введите пароль: 312231  
  
Повторите пароль: 123123  
  
Регистрируем...  
Пароли не совпадают!  
Пароль заполнен некорректно: Пароль должен содержать минимум 8 символов  
Почта некорректна  
Исправьте ошибки и попробуйте снова  
Хотите повторить регистрацию?  
Введите Y, если да и N, если нет: |
```

Рисунок 2. Функция регистрации пользователя

```
- Введите данные для регистрации -  
Имя: rrr  
Фамилия: ttt  
Email: Tit@bk.ru  
Введите пароль: Pro.1208339  
  
Повторите пароль: Pro.1208339  
  
Регистрируем...  
? УСПЕХ: Registered  
ID клиента: 1070  
ID кошелька: 1069  
Пароль: Pro.1208339  
  
Нажмите любую клавишу для продолжения...
```

Рисунок 3. Успешная регистрация пользователя

### 3. Вход пользователя (Рисунок 4, 5)

```
Выберите действие:  
1) Регистрация  
2) Вход  
0) Выход  
Ваш выбор: 2  
Email: test@gmail.com  
Пароль: 5Mn0lowN$Y8  
Неверный логин или пароль
```

Рисунок 4. Неуспешный вход в систему

```
Выберите действие:  
1) Регистрация  
2) Вход  
0) Выход  
Ваш выбор: 2  
Email: ff22@gmail.com  
Пароль: 12k3eIfyggT3  
Вход выполнен. Роль: Student
```

Рисунок 5. Успешный вход в аккаунт

### 4. Интерфейс для студента и просмотр доступных курсов (Рисунок 6, 7, 8)

```
===== Интерфейс Студента =====  
  
===== Информация о пользователе =====  
Имя: rrr  
Фамилия: ttt  
Почта: Tit@bk.ru  
Баланс: 0,00  
=====
```

- 1) Мои курсы
- 2) Доступные курсы
- 3) Бесплатные курсы
- 4) Записаться на курс
- 5) Отписаться от курса
- 6) Редактирование пользователя
- 7) Удалить аккаунт
- 8) Пополнить баланс
- 9) Просмотр баланса
- 10) Поиск курсов по названию курса
- 0) Выход

Ваш выбор: |

Рисунок 6. Интерфейс студента

```
ДОСТУПНЫЕ КУРСЫ  
-----  
  
1. JavaScript и React  
ID: 11  
Цена: 22000,00  
Дата: 10.03.24 - 10.06.24  
Статус: Не подписан  
  
2. Основы C#  
ID: 12  
Цена: 1999,99  
Дата: 15.10.25 - 29.10.25  
Статус: Не подписан  
  
Всего доступных курсов: 2  
  
Нажмите любую клавишу для продолжения...  
|
```

Рисунок 7. Доступные курсы

```

===== Информация о пользователе =====
Имя: rrr
Фамилия: ttt
Почта: Tit@bk.ru
Баланс: 0,00
=====

1) Мои курсы
2) Доступные курсы
3) Бесплатные курсы
4) Записаться на курс
5) Отписаться от курса
6) Редактирование пользователя
7) Удалить аккаунт
8) Пополнить баланс
9) Просмотр баланса
10) Поиск курсов по названию курса
0) Выход
Ваш выбор: 3
Нет доступных бесплатных курсов.

Нажмите любую клавишу для продолжения...

```

Рисунок 8. Вывод бесплатных курсов

#### 4. Запись студента на курсы (Рисунок 9, 10)

```

0) Выход
Ваш выбор: 4

ДОСТУПНЫЕ КУРСЫ
-----

1. JavaScript и React
ID: 11
Цена: 22000,00
Дата: 10.03.24 - 10.06.24
Статус: Не подписан

2. Основы C#
ID: 12
Цена: 1999,99
Дата: 15.10.25 - 29.10.25
Статус: Не подписан

Всего доступных курсов: 2

Нажмите любую клавишу для продолжения...
d
=====
Введите ID курса для записи: 11
Ошибка при записи на курс. Возможно, курс не найден или вы уже записаны на него, или недостаточно средств.

Нажмите любую клавишу для продолжения...

```

Рисунок 9. Неуспешная запись на курс

```
ДОСТУПНЫЕ КУРСЫ
-----

1. JavaScript и React
  ID: 11
  Цена: 22000,00
  Дата: 10.03.24 - 10.06.24
  Статус: Не подписан

2. Основы C#
  ID: 12
  Цена: 1999,99
  Дата: 15.10.25 - 29.10.25
  Статус: Не подписан

Всего доступных курсов: 2

Нажмите любую клавишу для продолжения...
1
=====
Введите ID курса для записи: 11
Вы подписались на курс!
На вашем балансе осталось => 101132,00

Нажмите любую клавишу для продолжения...
```

Рисунок 10. Успешная запись на курсы

4. Отписка от курса (Рисунок 11, 12)



```
=====
                        МОИ КУРСЫ
=====

КУРС #1
  Id курса: 11
  Название: JavaScript и React
  Цена: 22000,00
  Период: 10.03.24 - 10.06.24
  Статус: Завершен

=====
Всего курсов: 1
=====

Нажмите любую клавишу для продолжения...
d
=====
Введите ID курса для отписки: 11
Вы отписались от курса

Нажмите любую клавишу для продолжения...
```

Рисунок 11. Успешная отписка от курса

```
===== Интерфейс

===== Информация о пользователе =====
  Имя: rrr
  Фамилия: ttt
  Почта: Tit@bk.ru
  Баланс: 101132,00
=====

1) Мои курсы
2) Доступные курсы
3) Бесплатные курсы
4) Записаться на курс
5) Отписаться от курса
6) Редактирование пользователя
7) Удалить аккаунт
8) Пополнить баланс
9) Просмотр баланса
10) Поиск курсов по названию курса
0) Выход
Ваш выбор: 5
У вас нет записей на курсы.

Нажмите любую клавишу для продолжения...
```

Рисунок 12. Неуспешная отписка от курса

## 5. Обновление пароля (Рисунок 13, 14)

65	71	ff	ff	d20ff73a740982dfd16e3ad33c25f943167d5945d4f3fa1357b...	ff22@gmail.com	1	71
----	----	----	----	--	----------------	---	----

Рисунок 13. До обновления пароля

65	71	ff	ff	68a72c0d5547ba931225c01df232e29094e7b828f8d5b81f9...	ff22@gmail.com	1	71
----	----	----	----	--	----------------	---	----

```
=== СТУДЕНТ ===
1) Мои курсы
2) Доступные курсы
3) Бесплатные курсы
4) Записаться на курс
5) Отписаться от курса
6) Обновить пароль
7) Удалить аккаунт
8) Пополнить баланс
9) Просмотр баланса
10) Обновить почту (email) (Временно недоступно)
0) Выход
Ваш выбор: 6
Новый пароль: password2233
Пароль обновлён
```

Рисунок 14. После обновления пароля

## 6. Пополнение баланса и его просмотр (Рисунок 15, 16, 17)

```
=== СТУДЕНТ ===
1) Мои курсы
2) Доступные курсы
3) Бесплатные курсы
4) Записаться на курс
5) Отписаться от курса
6) Обновить пароль
7) Удалить аккаунт
8) Пополнить баланс
9) Просмотр баланса
10) Обновить почту (email) (Временно недоступно)
0) Выход
Ваш выбор: 8
Введите сумму для пополнения баланса:
Некорректная сумма
```

Рисунок 15. Некорректная сумма

```

===== Информация о пользователе =====
Имя: rrr
Фамилия: ttt
Почта: Tit@bk.ru
Баланс: 0,00
=====

1) Мои курсы
2) Доступные курсы
3) Бесплатные курсы
4) Записаться на курс
5) Отписаться от курса
6) Редактирование пользователя
7) Удалить аккаунт
8) Пополнить баланс
9) Просмотр баланса
10) Поиск курсов по названию курса
0) Выход
Ваш выбор: 8
Введите сумму для пополнения баланса: 123132
Кошелек пополнен

Нажмите любую клавишу для продолжения...

```

Рисунок 16. Успешное пополнение баланса

```

===== Информация о пользователе =====
Имя: rrr
Фамилия: ttt
Почта: Tit@bk.ru
Баланс: 101132,00
=====

```

Рисунок 17. Просмотр баланса

## 8. Удаление аккаунта

```

Точно удалить аккаунт? (y/n): y
Для того чтобы удалить аккаунт, введите пароль: Pго.1208339
Аккаунт удалён

Нажмите любую клавишу для продолжения...

```

Рисунок 18. Невозможное удаление аккаунта (Есть ещё связи)

## 9. Интерфейс Администратора (Рисунок 19, 20)

```

Выберите действие:
1) Регистрация
2) Вход
0) Выход
Ваш выбор: 2
Email: ff22@gmail.com
Пароль: aNmJRck8YEo@
Вход выполнен. Роль: Admin

```

Рисунок 19. Вход в аккаунт с ролью “Admin”

```

=== АДМИНИСТРАТОР ===
1) Изменить роль пользователю
2) Удалить курс
0) Выход

```

Рисунок 20. Интерфейс админа

#### 10. Изменение роли пользователя (Рисунок 21)

```

ID: 1030, Имя: Иван, Email: test638959622619470778@test.com, Роль: Student
ID: 1031, Имя: Петр, Email: test638959623403597083@test.com, Роль: Student
ID: 1033, Имя: Иван, Email: test638959623405629958@test.com, Роль: Student
ID: 1034, Имя: Петр, Email: test638959624844599296@test.com, Роль: Student
ID: 1036, Имя: Иван, Email: test638959624846486190@test.com, Роль: Student
ID: 1037, Имя: Петр, Email: test638959628047671042@test.com, Роль: Student
ID: 1039, Имя: Иван, Email: test638959628049528640@test.com, Роль: Student
ID: 1040, Имя: Петр, Email: test638959630371465326@test.com, Роль: Student
ID: 1042, Имя: Иван, Email: test638959630373026138@test.com, Роль: Student
ID: 1043, Имя: Петр, Email: test638959632395988667@test.com, Роль: Student
ID: 1045, Имя: Иван, Email: test638959632397986425@test.com, Роль: Student
ID: 1046, Имя: Петр, Email: test638959640018327411@test.com, Роль: Student
ID: 1048, Имя: Иван, Email: test638959640019920244@test.com, Роль: Student
ID: 1049, Имя: Петр, Email: test638960337839103935@test.com, Роль: Student
ID: 1051, Имя: Иван, Email: test638960337840452437@test.com, Роль: Banned

Введите ID пользователя: 1051

Список ролей:
ID: 2, Роль: Admin
ID: 4, Роль: Banned
ID: 1, Роль: Student
ID: 3, Роль: Teacher
Введите новую роль (ID): 3

```

Рисунок 21. Обновление роли

#### 11. Удаление курса (Рисунок 22, 23)

```
=== АДМИНИСТРАТОР ===
1) Изменить роль пользователю
2) Удалить курс
0) Выход
Ваш выбор: 2

Список курсов:
ID: 12, Название: Основы C#, Преподаватель ID: 1010

Введите ID курса для удаления: 100|
```

Рисунок 22. Неуспешное удаление курса

```
=== АДМИНИСТРАТОР ===
1) Изменить роль пользователю
2) Удалить курс
0) Выход
Ваш выбор: 2

Список курсов:
ID: 12, Название: Основы C#, Преподаватель ID: 1010

Введите ID курса для удаления: |
```

Рисунок 23. Успешное удаление курса

## 12. Интерфейс куратора (Рисунок 24, 25, 26)

```
Выберите действие:
1) Регистрация
2) Вход
0) Выход
Ваш выбор: 2
Email: ff22@gmail.com
Пароль: aNmJRck8YЕo@
Вход выполнен. Роль: Teacher
```

Рисунок 24. Успешный вход с ролью “Teacher”

```

=== ПРЕПОДАВАТЕЛЬ ===
1) Добавить курс
2) Мои курсы
3) Посмотреть записанных на курс
4) Удалить мой курс
0) Выход
Ваш выбор: |

```

Рисунок 25. Сам интерфейс

```

=== ПРЕПОДАВАТЕЛЬ ===
1) Добавить курс
2) Мои курсы
3) Посмотреть записанных на курс
4) Удалить мой курс
0) Выход
Ваш выбор: 2

- Ваши курсы -
ID: 19 | asdadsas | 2002-02-23 ? 2222-02-23 | 1 000,00 ?

Нажмите любую клавишу для продолжения...

```

Рисунок 26. Вывод курсов, которые принадлежат куратору

### 13. Просмотр записанных на курс (Рисунок 27, 28)

```

=== ПРЕПОДАВАТЕЛЬ ===
1) Добавить курс
2) Мои курсы
3) Посмотреть записанных на курс
4) Удалить мой курс
0) Выход
Ваш выбор: 3
Введите ID курса: 19
Либо курс не ваш, либо на курс пока никто не записан.

Нажмите любую клавишу для продолжения...

```

Рисунок 27. Неуспешный просмотр подписанных на курс

```

=== ПРЕПОДАВАТЕЛЬ ===
1) Мои курсы
2) Посмотреть записанных на курс
3) Удалить мой курс
0) Выход
Ваш выбор: 2
Введите ID курса: 15

- Записанные пользователи -
ID: 80 | qq qq | qq@gmail.com

```

Рисунок 28. Успешный просмотр

#### 14. Удаление курса (Рисунок 29, 30)

```
=== ПРЕПОДАВАТЕЛЬ ===  
1) Добавить курс  
2) Мои курсы  
3) Посмотреть записанных на курс  
4) Удалить мой курс  
0) Выход  
Ваш выбор: 4  
Введите ID курса для удаления: 111  
Не удалось удалить курс (возможно, это не ваш курс)  
Нажмите любую клавишу для продолжения...
```

Рисунок 29. Неуспешное удаление курса

```
=== ПРЕПОДАВАТЕЛЬ ===  
1) Добавить курс  
2) Мои курсы  
3) Посмотреть записанных на курс  
4) Удалить мой курс  
0) Выход  
Ваш выбор: 4  
Введите ID курса для удаления: 19  
Курс удален  
Нажмите любую клавишу для продолжения...
```

Рисунок 30. Успешное удаление курса

#### 14. Забаненный аккаунт (Рисунок 31)

```
Выберите действие:  
1) Регистрация  
2) Вход  
0) Выход  
Ваш выбор: 2  
Email: ff22@gmail.com  
Пароль: aNmJRck8YEo@  
Аккаунт был забанен
```

Рисунок 31. Забаненный пользователь

## Тестирование программы

### 1. Тест-кейс: Изменение роли пользователя администратором (Рисунок 32)

<b>Описание</b>
Функция позволяет администратору изменять роль пользователя в системе онлайн-обучения (роли: администратор, преподаватель, студент).
<b>Вводные данные</b>
<ol style="list-style-type: none"><li>1. Администратор авторизован в системе.</li><li>2. Пользователь существует в базе данных.</li><li>3. Выбрана одна из доступных ролей: "Администратор", "Преподаватель", "Студент".</li></ol>
<b>Обработка</b>
<ol style="list-style-type: none"><li>1. Администратор выбирает пользователя из списка.</li><li>2. Изменяет роль пользователя на одну из доступных.</li><li>3. Сохраняет изменения.</li><li>4. Система проверяет корректность данных и обновляет роль пользователя.</li></ol>
<b>Выходные данные</b>
<ol style="list-style-type: none"><li>1. При успешном изменении роли система отображает сообщение: "Роль пользователя успешно обновлена".</li><li>2. Выводит ID пользователя.</li><li>3. Номер новой роли.</li></ol>
<b>Сообщения об ошибках</b>
"Не удалось изменить роль пользователя"
<b>Требования к функционалу</b>
Функция доступна только пользователям с ролью "Администратор".

```
=====
Попытка обновить роль пользователя...
✅ Роль пользователя успешно обновлена
ID пользователя: 1067
Новая роль: 4
=====
Удален тестовый пользователь с ID: 1067
```

Рисунок 32. Тест-кейс изменения роли администратором



## 2. Тест-кейс: Регистрация нового студента (Рисунок 33)

<b>Описание</b>
Функция позволяет новому пользователю зарегистрироваться в системе онлайн-обучения в качестве студента для получения доступа к курсам и функционалу платформы.
<b>Вводные данные</b>
Пользователь не авторизован в системе Система доступна и функционирует нормально Доступ к интернету присутствует Корректные данные для регистрации
<b>Обработка</b>
Пользователь выбирает "Регистрация" из предложенных вариантов Система отображает форму регистрации с обязательными полями: Имя Фамилия Электронная почта Пароль (минимум 12) Пользователь заполняет все обязательные поля валидными данными Пользователь нажимает кнопку "Enter" Система проверяет корректность введенных данных Система создает новую учетную запись с ролью "Студент"
<b>Выходные данные</b>
"Вход выполнен. Роль: Student"
<b>Сообщения об ошибках</b>
"Не удалось зарегистрировать аккаунт"
<b>Требования к функционалу</b>
Функция доступна только не авторизованным пользователям Пароль должен содержать минимум 12 символов

```

=====
Тестирование регистрации с почтой test638963003127105740@test.com
Результат регистрации: UserID => 1069, Status => Registered
Результат не NULL
UserID => валидный: 1069
Status => 'Registered'
Сгенерированные пароли совпадают
Все тесты пройдены
=====
Удален тестовый пользователь с ID: 1069
=====

```

Рисунок 33. Регистрация студента в систему

### 3. Тест-кейс: Добавление нового курса преподавателем (Рисунок 34)

<b>Описание</b>
Функция позволяет преподавателю создавать и добавлять новые курсы в систему онлайн-обучения для дальнейшего обучения студентов.
<b>Вводные данные</b>
<ol style="list-style-type: none"> <li>1. Преподаватель авторизован в системе под своей учетной записью</li> <li>2. Преподаватель имеет роль "Teacher"</li> <li>3. Система доступна и функционирует нормально</li> <li>4. Доступ к интернету присутствует</li> </ol>
<b>Обработка</b>
<ol style="list-style-type: none"> <li>1. Нажимает кнопку "Добавить курс"</li> <li>2. Система отображает форму создания курса с полями: <ol style="list-style-type: none"> <li>а. Название курса</li> <li>б. Начало курса</li> <li>с. Окончание курса</li> <li>д. Стоимость курса</li> </ol> </li> <li>3. Преподаватель нажимает кнопку "Enter"</li> <li>4. Система проверяет корректность введенных данных</li> </ol>
<b>Выходные данные</b>
"Курс успешно создан"
<b>Сообщения об ошибках</b>
"Не удалось создать курс"
<b>Требования к функционалу</b>
<ul style="list-style-type: none"> <li>• Функция доступна только пользователям с ролью "Teacher"</li> <li>• Все обязательные поля должны быть заполнены</li> <li>• Требуется подключение к интернету для сохранения курса в базе данных</li> </ul>

```

=====
Попытка создать курс: 'Основы С#', даты 2025-10-18 - 2025-11-01, цена 1999,99
Результат создания курса: True
Курс успешно создан (ожидалось true)
=====
Удалено связанных курсов: 1
Удален тестовый пользователь с ID: 1065

```

Рисунок 34. Создание курса преподавателем

## **Выводы**

В ходе разработки системы онлайн-обучения были успешно достигнуты все поставленные цели и задачи. Проект реализован как стабильное, безопасное и производительное консольное приложение с эффективным использованием ADO.NET и LINQ для взаимодействия с базой данных.

### **Основные достижения:**

- Разработана нормализованная база данных, соответствующая 3NF, с продуманной структурой таблиц и связей
- Реализовано безопасное взаимодействие с БД с защитой от SQL-инъекций
- Внедрено хэширование паролей для обеспечения безопасности пользовательских данных
- Создан интуитивно понятный консольный интерфейс с разделением функционала по ролям

### **Трудности и их преодоление:**

#### **1. Проектирование базы данных:**

**Проблема:** Изначальное отсутствие таблицы для связи многие-ко-многим между пользователями и курсами

**Решение:** Создана таблица UserCourses, что позволило правильно организовать отношения и соответствовать 3NF

#### **2. Валидация данных:**

**Проблема:** Обеспечение целостности данных при вводе пользователем и их правильная обработка

**Решение:** Разработана система проверки вводимых данных на всех этапах работы приложения

#### **3. Управление зависимостями:**

**Проблема:** Обработка ситуаций при удалении записей с существующими связями

**Решение:** Реализована система проверки связанных данных перед выполнением операций удаления

**Результаты тестирования:**

Проведенное тестирование подтвердило корректность работы всех функций системы. Тест-кейсы для основных сценариев использования успешно пройдены, что демонстрирует надежность разработанного решения.

## Приложение

### Изменение роли в AdminService.cs

```
public async Task<bool> UpdateUserRoleAsync(int userId, int newRoleId)
{
    using var db = new AppDbContext();

    var userExists = await db.Users.AnyAsync(u => u.UserId == userId);

    if (!userExists)
        return false;

    try
    {
        var result = await db.Users.Where(u => u.UserId == userId).ExecuteUpdateAsync(setters =>
        setters.SetProperty(u => u.level_access_id, newRoleId));

        return result > 0;
    }

    catch (Exception ex) {

        Console.WriteLine($"Error updating user role: {ex.Message}");

        return false;
    }
}
```

## Реализации вывода в AdminView.cs

```
private async Task ChangeUserRoleAsync()
{
    using var context = new AppDbContext();

    // Простой вывод пользователей

    var users = await context.Users
        .Include(u => u.LevelAccess)
        .ToListAsync();

    Console.WriteLine("\nСписок пользователей:");

    foreach (var user in users)
    {
        Console.WriteLine($"ID: {user.UserId}, Имя: {user.first_name}, Email: {user.user_email}, Роль: {user.LevelAccess?.Title}");
    }

    Console.Write("\nВведите ID пользователя: ");

    if (!int.TryParse(Console.ReadLine(), out int userId))
    {
        Console.WriteLine("Некорректный ID пользователя");

        return;
    }

    // Простой вывод ролей

    var roles = await context.LevelAccess.ToListAsync();

    Console.WriteLine("\nСписок ролей:");

    foreach (var role in roles)
    {
        Console.WriteLine($"ID: {role.AccessId}, Роль: {role.Title}");
    }

    Console.Write("Введите новую роль (ID): ");
}
```

```

if (!int.TryParse(Console.ReadLine(), out int newRoleId))
{
    Console.WriteLine("Некорректный ID роли");
    return;
}

// Изменение роли
var userToUpdate = await context.Users.FindAsync(userId);
if (userToUpdate != null)
{
    userToUpdate.level_access_id = newRoleId;
    await context.SaveChangesAsync();

    Console.WriteLine("Роль пользователя успешно изменена!");
}
else
{
    Console.WriteLine("Пользователь не найден");
}
}

```

## Вывод информации в StudentService.cs

```
public async Task<List<object>> ShowinfoUser(int _userId)
{
    using var db = new AppDbContext();
    var checkouser = await db.Users.Where(u => u.UserId == _userId).Select(u => new
    {
        u.first_name,
        u.last_name,
        u.user_email,
        u.Wallet.balance
    }).ToListAsync();
    return checkouser.Cast<object>().ToList();
}
```



## Реализация вывода в StudentView.cs

```
private async Task ShowinfoUser(int _userId)
{
    try
    {
        Console.WriteLine("===== Информация о пользователе =====");
        var checkouser = await _studentService.ShowinfoUser(_userId);
        if (checkouser.Count == 0)
        {
            Console.WriteLine("Произошла ошибка вывода информации о пользователе");
        }
        foreach (dynamic info in checkouser)
        {
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine($"  Имя: {info.first_name}");
            Console.WriteLine($"  Фамилия: {info.last_name}");
            Console.WriteLine($"  Почта: {info.user_email}");
            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine($"  Баланс: {info.balance}");
            Console.ResetColor();
            Console.WriteLine("===== \n");
        }
    }
    catch (DbException dbex)
    {
        Console.WriteLine($"Ошибка БД -> {dbex.Message}");
    }
    catch (Exception ex)
    {
    }
```

```
{  
    Console.WriteLine($"Внезапная ошибка -> {ex.Message}");  
}  
}
```

## Создание курса в TeacherService.cs

```
public async Task<bool> CreateCourseAsync(int teacherId, string name, DateTime startDate, DateTime
endDate, decimal price)
{
    if (string.IsNullOrEmpty(name))
        return false;

    if (endDate < startDate)
        return false;

    if (price < 0)
        return false;

    using var db = new AppDbContext();

    var course = new Course
    {
        name_course = name.Trim(),
        data_start = startDate,
        data_end = endDate,
        price = price,
        UserID = teacherId
    };

    await db.Courses.AddAsync(course);

    await db.SaveChangesAsync();

    return true;
}
```

## Реализация в TeacherView.cs

```
private async Task CreateCourseAsync()
{
    Console.Write("Название курса: ");
```

```

var name = Console.ReadLine() ?? string.Empty;

Console.Write("Дата начала (yyyy-MM-dd): ");

if (!DateTime.TryParse(Console.ReadLine(), out DateTime startDate))
{
    Console.ForegroundColor = ConsoleColor.Red;

    Console.WriteLine("Некорректная дата начала");

    Console.ResetColor();

    Console.WriteLine("\nНажмите любую клавишу для продолжения...");

    Console.ReadKey();

    return;
}

Console.Write("Дата окончания (yyyy-MM-dd): ");

if (!DateTime.TryParse(Console.ReadLine(), out DateTime endDate))
{
    Console.ForegroundColor = ConsoleColor.Red;

    Console.WriteLine("Некорректная дата окончания");

    Console.ResetColor();

    Console.WriteLine("\nНажмите любую клавишу для продолжения...");

    Console.ReadKey();

    return;
}

Console.Write("Цена: ");

if (!decimal.TryParse(Console.ReadLine(), out decimal price))
{
    Console.ForegroundColor = ConsoleColor.Red;

    Console.WriteLine("Некорректная цена");

    Console.ResetColor();

    Console.WriteLine("\nНажмите любую клавишу для продолжения...");

    Console.ReadKey();
}

```

```
        return;
    }

    var ok = await _teacherService.CreateCourseAsync(_teacherId, name, startDate, endDate, price);

    if (ok)
    {
        Console.ForegroundColor = ConsoleColor.Green;

        Console.WriteLine("Курс добавлен");

        Console.ResetColor();
    }
    else
    {
        Console.ForegroundColor = ConsoleColor.Red;

        Console.WriteLine("Не удалось добавить курс. Проверьте данные.");

        Console.ResetColor();
    }

    Console.WriteLine("\nНажмите любую клавишу для продолжения...");

    Console.ReadKey();
}
```

## Реализация теста логирования в Tests.cs

```
public async Task TestLoginSuccess()
{
    Console.WriteLine();

    Console.WriteLine("=====");

    var authservice = new AuthService();

    var registrationservice = new UserRegistrationService();

    long createdUserId = 0;

    try
    {
        Console.WriteLine($"\"{33mРегистрация пользователя для теста логина: {testEmail}\"{0m}");

        await registrationservice.RegisterClientAsync("Петр", "Петров", testEmail, testPassword);

        Console.WriteLine($"\"{36mПопытка логина с email: {testEmail}\"{0m}");

        var result = await authservice.LoginAsync(testEmail, testPassword);

        createdUserId = result.UserId;

        Console.WriteLine($"\"{36mРезультат логина: Success => {result.Success}, UserId => {result.UserId}, Role => {result.RoleName}\"{0m}");

        Assert.True(result.Success);

        Console.WriteLine($"\"{32mSuccess => true\"{0m}");

        Assert.True(result.UserId > 0);

        Console.WriteLine($"\"{32mUserId => валидный: {result.UserId}\"{0m}");

        Assert.False(string.IsNullOrEmpty(result.RoleName));

        Console.WriteLine($"\"{32mRoleName => не пустой: {result.RoleName}\"{0m}");

        Console.WriteLine($"\"{92mТест на вход пройден\"{0m}");

        Console.WriteLine();

        Console.WriteLine("=====");
    }

    finally
    {

```

```
if (createdUserId > 0)
{
    await DeleteTestUser(createdUserId);
}
Console.WriteLine();
Console.WriteLine("=====");
}
}
```