

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

*Факультет Информационных технологий
Кафедра Информатики и информационных технологий*

направление подготовки

09.03.02 «Информационные системы и технологии»

КУРСОВОЙ ПРОЕКТ

Дисциплина: Технология кроссплатформенного программирования

Тема: Разработка на базе кроссплатформенных WEB технологий
информационной системы онлайн-библиотеки_____

Выполнил: студент группы 211-723

Сергеев Станислав Олегович

Дата, подпись: 18.12.2023

Проверил: _____

(Фамилия И.О., степень, звание)

Дата, подпись _____

(Дата)

(Подпись)

Замечания: _____

Москва

2023

Оглавление

ВВЕДЕНИЕ	3
ГЛАВА 1. Проектирование	4
1.1 Описание предметной области.....	4
1.2 Выбор инструментов.....	4
ГЛАВА 2. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ «YourLibrary»	7
2.1 Проектирование и физическая реализация базы данных на СУБД.....	7
2.2 Разработка программного продукта	15
ЗАКЛЮЧЕНИЕ.....	32
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....	33

ВВЕДЕНИЕ

Тема проекта – информационная система, предназначенная для осуществления процессов аренды и возврата книг в библиотеке.

В современных условиях библиотеки являются важнейшими учреждениями, предоставляющими доступ к культурным и научным ресурсам. Однако, проблемой может стать эффективность процесса аренды и возврата книг, которые часто проходят вручную. Это может вызвать проблемы ведения учета и использования информации, что затрудняет работу библиотечных работников и удерживает пользователей от новых посещений. Решением данной проблемы может стать информационная система, которая автоматизирует и оптимизирует процессы работы библиотеки

Цель – создание на базе кроссплатформенных WEB технологий информационной системы для определённой библиотечной организации для упрощения процессов аренды и возврата книг.

Исходя из поставленной цели, были сформированы следующие задачи:

1. Изучение предметной области;
2. Выбор и обоснование инструментов разработки;
3. Проектирование базы данных
4. Реализация физической модели данных.

ГЛАВА 1. Проектирование

1.1 Описание предметной области

Библиотеки уже не являются просто хранилищем книг, они предоставляют доступ к ценным знаниям и информационным ресурсам. Однако, в процессе аренды и возврата книг могут возникать проблемы, которые затрудняют работу библиотеки. Это включает в себя учет книг, организацию работы библиотечных служащих и удовлетворение потребностей пользователей. Для решения этих проблем можно использовать информационную систему, которая автоматизирует и оптимизирует процессы аренды и возврата книг, что увеличит эффективность работы библиотеки и улучшит пользовательский опыт.

В современном мире, где доступ к знаниям является ключевым фактором успеха, библиотеки играют важную роль в обеспечении образования и культуры. Однако, процессы аренды и возврата книг в библиотеках часто проходят вручную, что приводит к сложностям ведения учета и организации работы библиотечных служащих. Решением этой проблемы может стать создание информационной системы, которая автоматизирует и оптимизирует процессы аренды и возврата книг, что облегчит работу библиотечной команды и повысит качество обслуживания пользователей.

1.2 Выбор инструментов

Выбор инструментов разработки информационной системы YourLibrary был сделан с учетом требований, которые предъявляются к серьезным проектам такого рода. Одним из ключевых элементов этого выбора стала Microsoft Visual Studio - мощная интегрированная среда разработки программного обеспечения, которая предоставляет разработчикам инструменты для создания высокопроизводительных приложений, удобных инструментов для отладки и тестирования, а также автоматизированных средств сборки и развертывания.

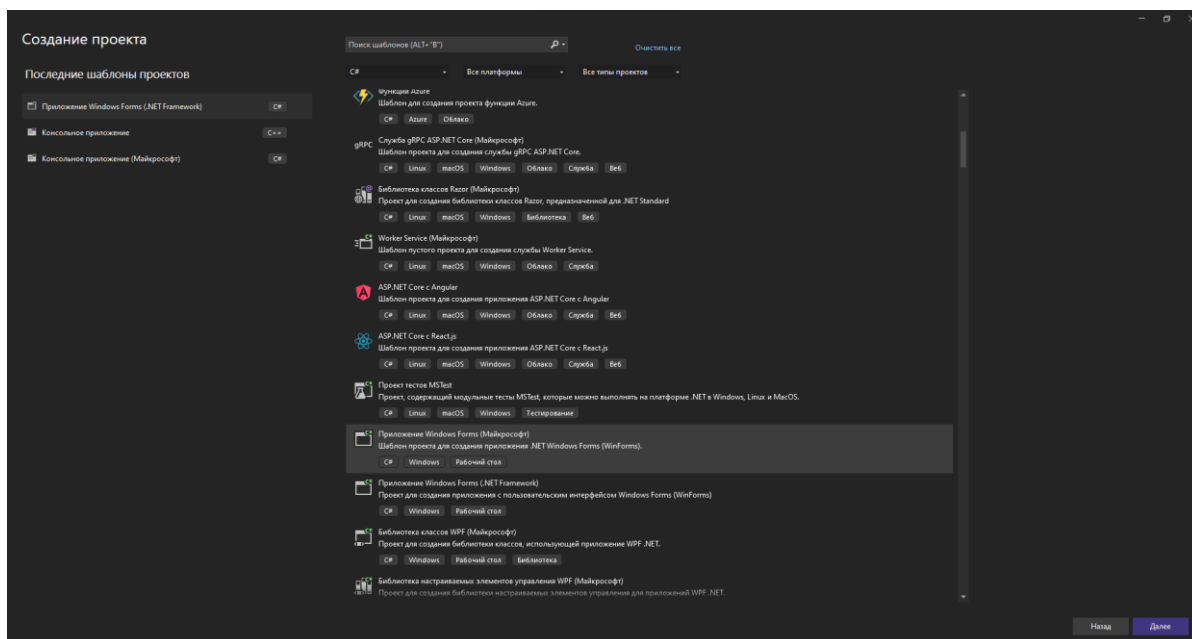


Рисунок 1.1 Оболочка Microsoft Visual Studio

Главным языком программирования, который был выбран для разработки YourLibrary, стал C# - мощный объектно-ориентированный язык программирования, который предоставляет наполненную библиотеку классов, богатые функциональные возможности и высокую производительность. С языком C# очень легко работать, поскольку он имеет простой и понятный синтаксис, что позволяет программистам сосредоточиться не на деталях языка, а на решении бизнес-задачи, которую они ставят перед своим проектом.

MS SQL (Microsoft SQL Server) является одной из самых популярных систем управления базами данных (СУБД) в мире. Ее выбор обосновывается несколькими факторами.

- 1) MS SQL обладает высокой производительностью и масштабируемостью. Эта СУБД способна обрабатывать большие объемы данных и поддерживать одновременную работу множества пользователей. Встроенные инструменты оптимизации запросов и индексирования помогают улучшить производительность работы с базой данных.

2) MS SQL обладает широким функционалом и поддерживает различные режимы работы. Она поддерживает транзакционную обработку данных, поддержку хранимых процедур и функций, а также предлагает механизмы репликации и кластеризации для обеспечения отказоустойчивости.

3) MS SQL предлагает удобную и мощную среду для разработки и администрирования баз данных. Она интегрируется с другими продуктами Microsoft, такими как Visual Studio, что облегчает разработку приложений с использованием базы данных. Богатый пользовательский интерфейс позволяет удобно взаимодействовать с базой данных и проводить различные операции, такие как создание таблиц, индексов и запросов.

Все эти факторы делают MS SQL привлекательным выбором для организаций, которым требуется мощная и надежная СУБД для обработки и управления большими объемами данных.

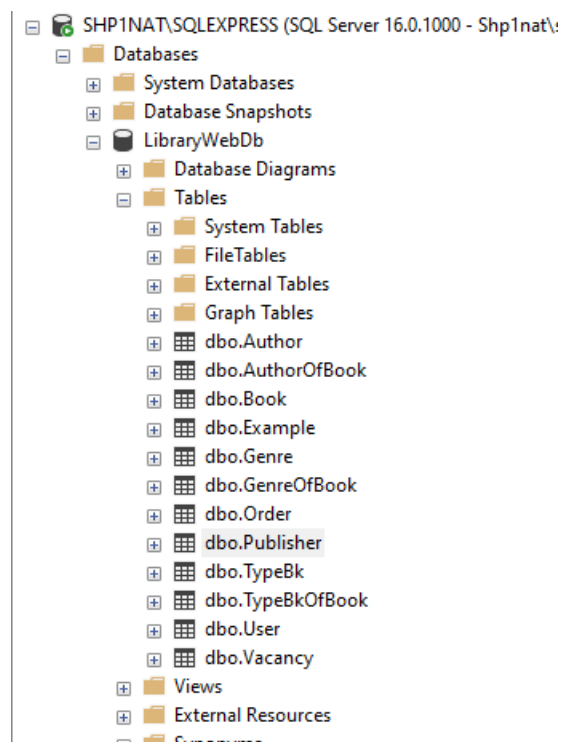


Рисунок 1.2 Оболочка MS SQL

ГЛАВА 2. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ «YourLibrary»

2.1 Проектирование и физическая реализация базы данных на СУБД

Для реализации данного проекта была создана база данных, состоящая из тринадцати таблиц. Таблица реляционной базы данных — это совокупность связанных данных, хранящихся в структурированном виде в базе данных. Структура и связи этой базы данных представлены на рисунке ниже.

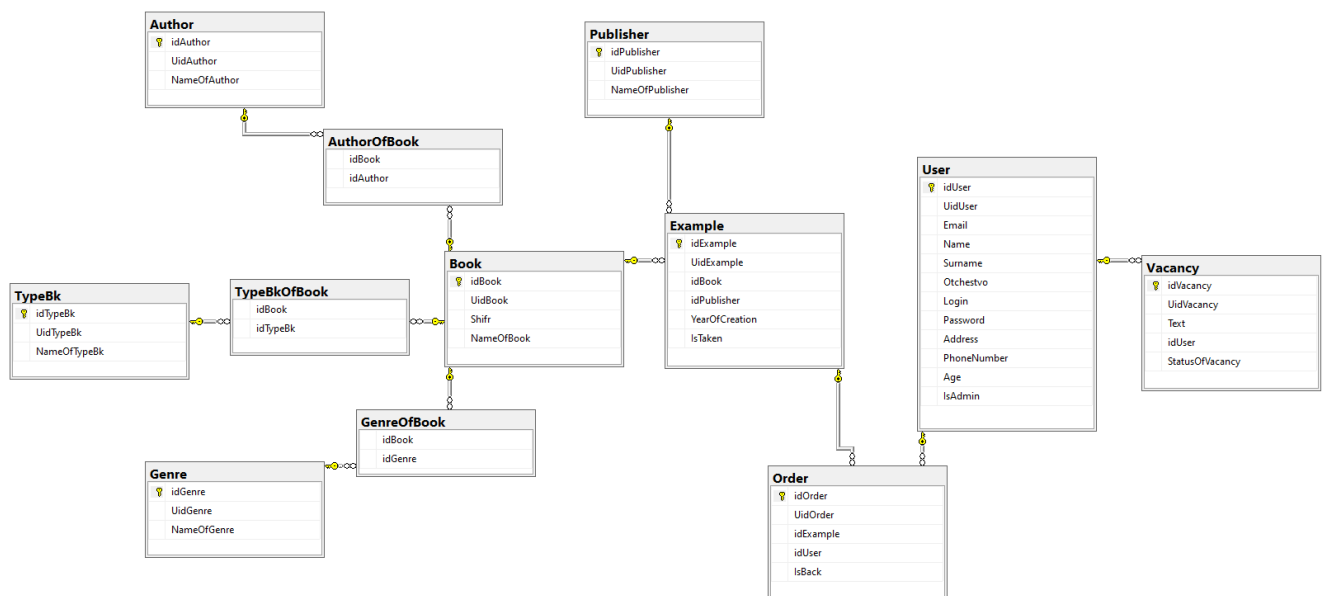


Рисунок 2.1 Структура базы данных

Главной таблицей, по которой сортируются книги является таблица Example, которая подразумевает единичный экземпляр книги. Таблица имеет связи с другими вспомогательными таблицами для определения атрибутов экземпляра: таблица Book, таблица Publisher. Таким образом, таблица Example будет обладать данными о названии книги, ее авторах, ее типах, ее жанрах, её издании. Вспомогательные таблицы используются для корректной систематизации книг в библиотеке. Например, одна и та же книга может быть

опубликована разными изданиями в разные годы. Далее будут представлены структуры всех таблиц с пояснением к каждой.

- 1) Book с колонками idBook INT, UidBook UNIQUEIDENTIFIER, Shifr NVARCHAR(256), NameOfBook NVARCHAR(256)

	Имя	Тип данных	Допустимы значения NULL	По умолчанию
PK	idBook	int	<input type="checkbox"/>	
	UidBook	uniqueidentifier	<input type="checkbox"/>	
	Shifr	nvarchar(256)	<input type="checkbox"/>	
	NameOfBook	nvarchar(256)	<input type="checkbox"/>	
			<input type="checkbox"/>	

Рисунок 2.2 Таблица книги

В таблице каждой книге присваивается шифр, название и уникальный айди. Выбор AI в колонке idBook подразумевает, что у каждой книги будет ее уникальный id, который будет автоматически присваиваться каждой книге.

Выбор NN в колонках idBook, Shift, NameOfBook подразумевает, что колонки таблицы не могут быть равны null (не могут быть незаполненными) при добавлении новой книги. Выбор PK в колонке idBook подразумевает, что колонка будет обладать статусом Primary key. Нужно это для того, чтобы связывать эту таблицу с другими по этой колонке.

- 2) Genre с колонками idGenre INT, UidGenre UNIQUEIDENTIFIER, NameOfGenre NVARCHAR(256)

	Имя	Тип данных	Допустимы значения NULL	По умолчанию
PK	idGenre	int	<input type="checkbox"/>	
	UidGenre	uniqueidentifier	<input type="checkbox"/>	
	NameOfGenre	nvarchar(256)	<input type="checkbox"/>	
			<input type="checkbox"/>	

Рисунок 2.3 Таблица жанра

В таблице каждому жанру присваивается имя и уникальный айди. Выбор AI в колонке idGenre подразумевает, что у каждого ряда будет его уникальный id, который будет автоматически присваиваться каждому ряду. Выбор NN в

колонках idGenre, NameOfGenre подразумевает, что колонки таблицы не могут быть равны null (не могут быть незаполненными) при добавлении нового ряда. Выбор PK в колонке idGenre подразумевает, что колонка будет обладать статусом Primary key. Нужно это для того, чтобы связывать эту таблицу с другими по этой колонке.

- 3) GenreOfBook с колонками idBook INT, idGenre INT. Эта таблица связывает таблицы book и genre

Имя	Тип данных	Допустимы значения NULL	По умолчанию
idBook	int	<input type="checkbox"/>	
idGenre	int	<input type="checkbox"/>	
		<input type="checkbox"/>	

Рисунок 2.4 Таблица связи книги и жанра

Это связующая таблица. Она связывает книгу и жанр по их айди. Таблица использует Primary Keys из таблиц idBook и idGenre, чтобы сопоставить связь между этими таблицами. Колонки не должны быть равны null, из-за чего мы выбираем NN, а также выбираем PK для использования Primary Keys.

- 4) Typebk с колонками idTypeBk INT, UidGenre UNIQUEIDENTIFIER, NameOfType NVARCHAR(256)


Имя	Тип данных	Допустимы значения NULL	По умолчанию
 idTypeBk	int	<input type="checkbox"/>	
UidTypeBk	uniqueidentifier	<input type="checkbox"/>	
NameOfTypeBk	varchar(256)	<input type="checkbox"/>	
		<input type="checkbox"/>	

Рисунок 2.5 Таблица типа

В таблице каждому типу присваивается имя и уникальный айди. Выбор AI в колонке idTypeBk подразумевает, что у каждого ряда будет его уникальный id, который будет автоматически присваиваться каждому ряду. Выбор NN в колонках idTypeBk, NameOfType подразумевает, что колонки таблицы не могут быть равны null (не могут быть незаполненными) при добавлении нового ряда.

Выбор PK в колонке idTypeBk подразумевает, что колонка будет обладать статусом Primary key. Это нужно для связи с другими таблицами.

- 5) TypeOfBook с колонками idTypeBk INT, idBook INT. Эта таблица связывает таблицы book и typebk

Имя	Тип данных	Допустимы значения NULL	По умолчанию
idBook	int	<input type="checkbox"/>	
idTypeBk	int	<input type="checkbox"/>	
		<input type="checkbox"/>	

Рисунок 2.6 Таблица связи книги и типа

Это связующая таблица. Она связывает книгу и тип по их айди. Таблица использует Primary Keys из таблиц idTypeBk и idBook, чтобы сопоставить связь этих таблицами. Колонки не должны быть равны null, из-за чего мы выбираем NN, выбираем PK для использования Primary Keys.

- 6) Author с колонками idAuthor INT, UidAuthor UNIQUEIDENTIFIER, NameOfAuthor NVARCHAR(256)

Имя	Тип данных	Допустимы значения NULL	По умолчанию
idAuthor	int	<input type="checkbox"/>	
UidAuthor	uniqueidentifier	<input type="checkbox"/>	
NameOfAuthor	nvarchar(256)	<input type="checkbox"/>	
		<input type="checkbox"/>	

Рисунок 2.7 Таблица автора

В таблице каждому автору присваивается имя и уникальный айди. Выбор AI в колонке idAuthor подразумевает, что у каждого ряда будет его уникальный id, который будет автоматически присваиваться каждому ряду. Выбор NN в колонках idAuthor, NameOfAuthor подразумевает, что колонки таблицы не могут быть равны null (не могут быть незаполненными) при добавлении нового ряда. Выбор PK в колонке idAuthor подразумевает, что колонка будет обладать статусом Primary key. Нужно это для того, чтобы связывать эту таблицу с другими по этой колонке.

- 7) Authorofbook с колонками idAuthor INT, idBook INT. Эта таблица связывает таблицы book и author

Имя	Тип данных	Допустимы значения NULL	По умолчанию
idBook	int	<input type="checkbox"/>	
idAuthor	int	<input type="checkbox"/>	
		<input type="checkbox"/>	

Рисунок 2.8 Таблица связи книги и автора

Это связующая таблица. Она связывает книгу и автора по их айди. Таблица использует Primary Keys из таблиц idBook и idAuthor, чтобы сопоставить связь между этими таблицами. Колонки не должны быть равны null, из-за чего мы выбираем NN, а также выбираем PK для использования Primary Keys. Наличие такой таблицы говорит о том, что у каждой книги может быть несколько авторов.

- 8) Example с колонками idExample INT, UidExample UNIQUEIDENTIFIER, idBook INT, idYearBk INT, idPublisher INT, isTaken TINYINT

Имя	Тип данных	Допустимы значения NULL	По умолчанию
idExample	int	<input type="checkbox"/>	
UidExample	uniqueidentifier	<input type="checkbox"/>	
idBook	int	<input type="checkbox"/>	
idPublisher	int	<input type="checkbox"/>	
YearOfCreation	int	<input type="checkbox"/>	
IsTaken	bit	<input checked="" type="checkbox"/>	0
		<input type="checkbox"/>	

Рисунок 2.9 Таблица экземпляра

В таблице каждому экземпляру присваивается книга из таблицы book по idBook, год издания из таблицы yearBk по idYearBk, издание книги из таблицы publisher по idPublisher, уникальный айди каждого экземпляра, а также значение isTaken: либо 0, если книга не взята в аренду и 1, если книга взята в аренду. Таблица использует Primary Keys из таблиц idBook, idYearBk, idPublisher чтобы сопоставить связь между этими таблицами. Колонки не должны быть равны

null, из-за чего мы выбираем NN, а также выбираем PK в колонке idExample для использования Primary Key для последующей связи этой таблицы с таблицей Order.

- 9) Publisher с колонками idPublisher INT, UidPublisher UNIQUEIDENTIFIER, NameOfPublisher NVARCHAR(256).

	Имя	Тип данных	Допустимы значения NULL	По умолчанию
PK	idPublisher	int	<input type="checkbox"/>	
	UidPublisher	uniqueidentifier	<input type="checkbox"/>	
	NameOfPublisher	nvarchar(256)	<input type="checkbox"/>	
			<input type="checkbox"/>	

Рисунок 2.10 Таблица издания

В таблице каждому изданию присваивается имя и уникальный айди. Выбор AI в колонке idPublisher подразумевает, что у каждого ряда будет его уникальный id, который будет автоматически присваиваться каждому ряду. Выбор NN в колонках idPublisher, NameOfPublisher подразумевает, что колонки таблицы не могут быть равны null (не могут быть незаполненными) при добавлении нового ряда. Выбор PK в колонке idPublisher подразумевает, что колонка будет обладать статусом Primary key. Нужно это для того, чтобы связывать эту таблицу с другими по этой колонке.

- 10) Order с колонками idOrder INT, UidOrder UNIQUEIDENTIFIER, idUser INT, idExample INT, isBack TINYINT

	Имя	Тип данных	Допустимы значения NULL	По умолчанию
PK	idOrder	int	<input type="checkbox"/>	
	UidOrder	uniqueidentifier	<input type="checkbox"/>	
	idExample	int	<input type="checkbox"/>	
	idUser	int	<input type="checkbox"/>	
	IsBack	bit	<input type="checkbox"/>	
			<input type="checkbox"/>	

Рисунок 2.11 Таблица заказа

В таблице каждому заказу присваивается уникальный айди, пользователь из таблицы user по idUser, экземпляр книги из таблицы example по idExample, а также колонка isBack, которая принимает 1, если аренда была завершена и 0, если заказ еще не закрыт (книга не возвращена).

- 11) User с колонками idUser INT, UidUser UNIQUEIDENTIFIER, Name NVARCHAR(256), Surname VARCHAR(256), Otchestvo NVARCHAR(256), Login NVARCHAR(256), Password NVARCHAR(256), Address NVARCHAR(256), PhoneNumber NVARCHAR(256), IsAdmin BIT

	Имя	Тип данных	Допустимы значения NULL	По умолчанию
№	idUser	int	<input type="checkbox"/>	
	UidUser	uniqueidentifier	<input type="checkbox"/>	
	Email	nvarchar(256)	<input type="checkbox"/>	
	Name	nvarchar(256)	<input checked="" type="checkbox"/>	
	Surname	nvarchar(256)	<input checked="" type="checkbox"/>	
	Otchestvo	nvarchar(256)	<input checked="" type="checkbox"/>	
	Login	nvarchar(256)	<input type="checkbox"/>	
	Password	nvarchar(256)	<input type="checkbox"/>	
	Address	nvarchar(256)	<input checked="" type="checkbox"/>	
	PhoneNumber	nvarchar(256)	<input checked="" type="checkbox"/>	
	Age	int	<input checked="" type="checkbox"/>	
	IsAdmin	bit	<input type="checkbox"/>	
			<input type="checkbox"/>	

Рисунок 2.12 Таблица пользователя

В таблице каждому пользователю уникальный айди, имя, фамилия, отчество(может быть null), логин, пароль, адрес(может быть null), номер телефона(может быть null) и статус админа. 0 – если это обычный пользователь, 1 – если пользователь обладает правами админа (сможет заходить в админский интерфейс, в котором можно будет управлять системой с точки зрения админа). Выбор AI в колонке idUser подразумевает, что у каждого ряда будет его уникальный id, который будет автоматически присваиваться каждому ряду.

Выбор NN в колонках idUser, Name, Surname, Login, Password подразумевает, что колонки таблицы не могут быть равны null (не могут быть незаполненными) при добавлении нового ряда. Выбор PK в колонке idUser подразумевает, что колонка будет обладать статусом Primary key. Нужно это для того, чтобы связывать эту таблицу с другими по этой колонке. В последствии эта таблица будет иметь связь с таблицами Vakansiya и Order. Это нужно, чтобы мы могли определить пользователей по их idUser, которые имеют какие-либо заказы или которые имеют какие-либо вакансии.

13) Vakansiya с колонками idVakansiya INT, UiVacancy UNIQUEIDENTIFIER, Text NVARCHAR(MAX), idUser INT, StatusOfVacancy INT

	Имя	Тип данных	Допустимы значения NULL	По умолчанию
№	idVacancy	int	<input type="checkbox"/>	
	UidVacancy	uniqueidentifier	<input type="checkbox"/>	
	Text	nvarchar(MAX)	<input checked="" type="checkbox"/>	
	idUser	int	<input type="checkbox"/>	
	StatusOfVacancy	int	<input checked="" type="checkbox"/>	0
			<input type="checkbox"/>	

Рисунок 2.13 Таблица вакансии

В таблице каждой вакансии присваивается уникальный айди, текст вакансии, айди юзера, который ее отправил и статус (0 – на рассмотрении, 1 – отклонена, 2 – принята, если 2 – то пользователю выдаются права админа). Таблица использует Primary Key из таблицы idUser чтобы сопоставить связь между этими таблицами. Колонки idVakansiya, idUser не должны быть равны null, из-за чего мы выбираем NN, а также выбираем PK в колонке idVakansiya для использования Primary Key для последующей связи этой таблицы с таблицей idUser. Текст вакансии сохраняется в колонке Text, которая имеет тип данных LONGTEXT, чтобы иметь возможность сохранять большие тексты информации.

На этом этап проектирования и физической реализации базы данных на СУБД заканчивается. Следующим этап будет служить этап программной реализации приложения в Visual Studio с использованием языка программирования C#, а также с использованием выбранной СУБД – MySQL Workbench.

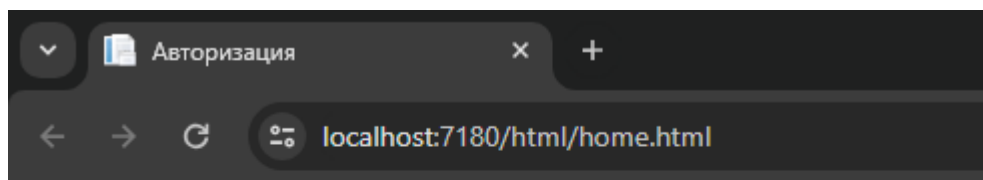
2.2 Разработка программного продукта

```
public Guid? Register(RegisterData registerData)
{
    var checkUser1 = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.User>().SingleOrDefault(x => x.Login == registerData.Login);
    var checkUser2 = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.User>().SingleOrDefault(x => x.Email == registerData.Email);
    if (checkUser1 != null || checkUser2 != null) { return null; }
    var user = new DatabaseAccessLayer.Entities.User
    {
        UidUser = Guid.NewGuid(),
        Email = registerData.Email,
        Login = registerData.Login,
        Password = GetHash(registerData.Password),
        IsAdmin = false
    };
    _libraryWebDbContext.Add(user);
    _libraryWebDbContext.SaveChanges();
    return user.UidUser;
}

Ссылка 1
public Guid? Login(LoginData loginData)
{
    var hashedPassword = GetHash(loginData.Password);
    var user = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.User>().SingleOrDefault(x => x.Email == loginData.Email && x.Password == hashedPassword);
    return user?.UidUser;
}
```

Рисунок 2.14 Код входа пользователя в систему

На рисунке продемонстрирован код подключения к базе данных, после чего вызывается функции Register и Login, которая проверяет пользователя на наличие его в таблице пользователя. Если он существует, то идет проверка его админ статуса. Если пользователь обладает таким, то ему предоставляется токен входа администратора. Если пользователь не имеет статуса админа, то он автоматически зайдет с пользовательским токеном. Данные о текущем пользователе (токен) имеют время жизни 180 минут.



Авторизация

Email: Пароль:

Рисунок 2.15 Интерфейс входа пользователя в систему

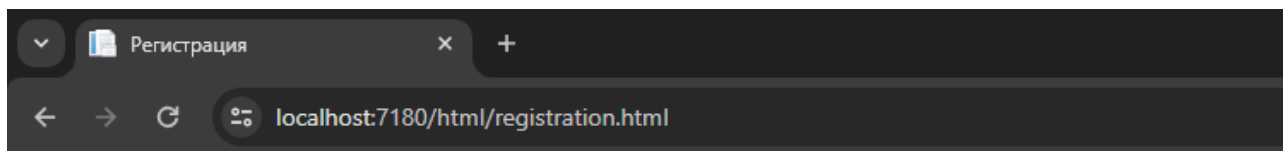
На рисунке продемонстрирован интерфейс входа пользователя в систему, а также представлен интерфейс кнопок для реализации выполнения кода по нажатию на эти кнопки. Имеющийся функционал:

- 1) Кнопка вход пользователя;
- 2) Кнопка вход админа;
- 3) Кнопка регистрации;
- 4) Текстовое поле для ввода логина;
- 5) Текстовое поле для ввода пароля;

```
public Guid? Register(RegisterData registerData)
{
    var checkUser1 = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.User>().SingleOrDefault(x => x.Login == registerData.Login);
    var checkUser2 = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.User>().SingleOrDefault(x => x.Email == registerData.Email);
    if (checkUser1 != null || checkUser2 != null) { return null; }
    var user = new DatabaseAccessLayer.Entities.User
    {
        UidUser = Guid.NewGuid(),
        Email = registerData.Email,
        Login = registerData.Login,
        Password = GetHash(registerData.Password),
        IsAdmin = false
    };
    _libraryWebDbContext.Add(user);
    _libraryWebDbContext.SaveChanges();
    return user.UidUser;
}
```

Рисунок 2.16 Код регистрации пользователя в системе

На рисунке продемонстрирован код подключения к базе данных, после чего вызывается функция `isLoginExist`, которая проверяет введенный пользователем логин на уже существующий. Если логин не занят, то вызывается процедура `AddNewUser`, которая регистрирует пользователя в системе (добавляет его в таблицу пользователя).



Регистрация

Login: Password: Email:

Рисунок 2.17 Интерфейс регистрации пользователя в системе

На рисунке продемонстрирован интерфейс регистрации пользователя в системе, а также представлен интерфейс кнопок для реализации выполнения кода по нажатию на эти кнопки. Имеющийся функционал:

- 1) Кнопка регистрации пользователя;
- 2) Кнопка возвращения;
- 3) Текстовое поле для ввода логина;
- 4) Текстовое поле для ввода пароля;

```

public List<Contract.Example>? GetExamples()
{
    var examples = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Example>()
        .Include(x => x.Book)
        .Include(x => x.Publisher)
        .Include(x => x.Book.AuthorsOfBook)
        .Include(x => x.Book.GenresOfBook)
        .Include(x => x.Book.TypesBkOfBook)
        .ToList();
    if (examples.Count == 0) return null;

    return examples.Select(example => new Contract.Example
    {
        UidExample = example.UidExample,
        Book = new Contract.Book
        {
            UidBook = example.Book.UidBook,
            Shifr = example.Book.Shifr,
            NameOfBook = example.Book.NameOfBook,
            Authors = example.Book.AuthorsOfBook.Select(x => x.NameOfAuthor).ToList(),
            Genres = example.Book.GenresOfBook.Select(x => x.NameOfGenre).ToList(),
            TypesBk = example.Book.TypesBkOfBook.Select(x => x.NameOfTypeBk).ToList()
        },
        Publisher = new Contract.Publisher
        {
            UidPublisher = example.Publisher.UidPublisher,
            NameOfPublisher = example.Publisher.NameOfPublisher
        },
        YearOfCreation = example.YearOfCreation,
        IsTaken = example.IsTaken
    }).ToList();
}

```

Рисунок 2.18 Код вывода всех доступных экземпляров

На рисунке продемонстрирован код подключения к базе данных, после чего вызывается процедура `GetAvailableExamples`, которая формирует таблицу доступных книг (у которых в колонке `isTaken` стоит 0).

guid	шифр	название	автор	жанр	тип	издательство	год выпуска	статус	Выбрать
a97bbee3-7fe3-40bb-a190-477c27ed33bd	string	string	string, Zxc Sda AAaw	string	string	MyFirstPublisher	2010	Доступно	<input type="checkbox"/>
a7086f39-f8c1-4920-835d-4d169cef131c	string	string	string, Zxc Sda AAaw	string	string	MyFirstPublisher	2005	Доступно	<input type="checkbox"/>
1e9e66e0-a2ba-44f9-826c-9b257f042b58	string	string	string, Zxc Sda AAaw	string	string	MyFirstPublisher	2007	Доступно	<input type="checkbox"/>
b71c32d3-cbab-46a0-8993-d7bce21d008c	string	string	string, Zxc Sda AAaw	string	string	MyFirstPublisher	2012	Доступно	<input type="checkbox"/>
01025405-9f24-48b1-abac-2a44540ed455	string	string	string, Zxc Sda AAaw	string	string	MyFirstPublisher	2015	Доступно	<input type="checkbox"/>
00ccf050-28fd-4033-bc5e-093b11880431	string	string	string, Zxc Sda AAaw	string	string	MyFirstPublisher	2020	Доступно	<input type="checkbox"/>
af5985db-1f57-4aa3-9754-01fa45254acd	string	string	string, Zxc Sda AAaw	string	string	MyFirstPublisher	2014	Доступно	<input type="checkbox"/>

Рисунок 2.19 Интерфейс вывода всех доступных экземпляров

На рисунке продемонстрирован интерфейс заказа пользователем книг, а также представлен интерфейс кнопок выполнения кода по нажатию на эти кнопки. Имеющийся функционал:

- 1) Кнопки пользовательского меню;
- 2) Кнопка заказать;
- 3) Возможность выбора нескольких рядов из таблицы книг.

```
public Guid? CreateOrder(Guid uIdExample, Guid uIdUser)
{
    var example = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Example>()
        .SingleOrDefault(x => x.UidExample == uIdExample);
    var user = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.User>()
        .SingleOrDefault(x => x.UidUser == uIdUser);
    if (example == null || example.IsTaken || user == null) return null;
    var order = new DatabaseAccessLayer.Entities.Order
    {
        UidOrder = Guid.NewGuid(),
        Example = example,
        User = user,
        IsBack = false
    };
    example.IsTaken = true;
    _libraryWebDbContext.Add(order);
    _libraryWebDbContext.SaveChanges();
    return order.UidOrder;
}
```

Рисунок 2.20 Код оформления аренды для пользователя

На рисунке продемонстрирован код подключения к базе данных, после чего с помощью запроса можно получить idUser человека по переменной systemLogin. Далее мы вызываем процедуру ToOrderTheBook с аргументами айди пользователя и айди экземпляра(ов), который(ые) пользователь выбрал. Эта процедура добавляет новый заказ в таблицу order и отмечает книгу, как взятую.

Все книги

Оформить заказ									
guid	шифр	название	автор	жанр	тип	издательство	год выпуска	статус	Выбрать
a97bbe3-7fe3-40bb-a190-477c27ed33bd	string	string	string, Zxc Sda AAw	string	string	MyFirstPublisher	2010	Доступно	<input checked="" type="checkbox"/>
a7086f39-f8c1-4920-855d-4d169cef131c	string	string	string, Zxc Sda AAw	string	string	MyFirstPublisher	2005	Доступно	<input type="checkbox"/>
1e9e66e0-a2ba-44f9-826c-9b257f042b58	string	string	string, Zxc Sda AAw	string	string	MyFirstPublisher	2007	Доступно	<input checked="" type="checkbox"/>
b71c32d3-cbab-46a0-8993-d7bce21d008c	string	string	string, Zxc Sda AAw	string	string	MyFirstPublisher	2012	Доступно	<input checked="" type="checkbox"/>
01025405-9f24-48b1-abac-2a44540ed455	string	string	string, Zxc Sda AAw	string	string	MyFirstPublisher	2015	Доступно	<input checked="" type="checkbox"/>
00ccf050-28fd-4033-bc5e-093b11880431	string	string	string, Zxc Sda AAw	string	string	MyFirstPublisher	2020	Доступно	<input type="checkbox"/>
af5985db-1f57-4aa3-9754-01fa45254acd	string	string	string, Zxc Sda AAw	string	string	MyFirstPublisher	2014	Доступно	<input type="checkbox"/>

Рисунок 2.21 Интерфейс выделения книг для заказа

На рисунке продемонстрирован интерфейс выделения пользователем книг, а также представлен интерфейс кнопок выполнения кода по нажатию на эти кнопки. Имеющийся функционал:

- 1) Кнопка заказать;
- 2) Возможность выбора нескольких рядов из таблицы книг.
- 3) Таблица доступных книг, которая имеет колонки: название, автор, жанры, тип, шифр, название, год

```
public bool CloseOrder(Guid uIdOrder)
{
    var order = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Order>()
        .Include(x => x.Example)
        .SingleOrDefault(x => x.UidOrder == uIdOrder);
    if (order == null) return false;
    order.Example.IsTaken = false;
    order.IsBack = true;
    _libraryWebDbContext.SaveChanges();
    return true;
}
```

```

async function closeOrders() {
  const checkboxes = document.querySelectorAll('#ordersTable tbody input[type="checkbox"]');
  const selectedOrders = [];
  if (checkboxes.length === 0) {
    alert('Выберите хотя бы один заказ');
    return;
  }
  checkboxes.forEach((checkbox, index) => {
    if (checkbox.checked) {
      const row = checkbox.closest('tr');
      const guid = row.querySelector('td:nth-child(1)').textContent;
      selectedOrders.push(guid);
    }
  });

  selectedOrders.forEach(async (uid) => {
    try {
      const response = await fetch(`https://localhost:7180/api/Order/CloseOrder?uIdOrder=${uid}`, {
        method: 'PUT',
        headers: {
          'Content-Type': 'application/json',
          Authorization: `Bearer ${localStorage.getItem('userToken')}`
        },
      });
      if (response.ok) {
      }
      else {
        throw new Error('Что-то пошло не так');
      }
    } catch (error) {
      console.error();
    }
  });
  location.reload(true);
}

```

Рисунок 2.22 Код админа для отметки возврата книг(и)

На рисунке продемонстрирован код админского подтверждения возврата книги. Продемонстрирован код подключения к базе данных, после чего вызывается процедура возврата книги, которая подтверждает ее возврат (закрывает заказ, после чего он появляется в истории заказов у пользователя, который оформлял его).

Все заказы

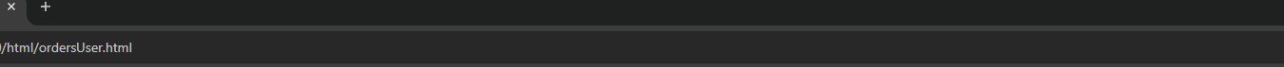
Заккрыть выбранные заказы											
Guid заказа	Guid экземпляра	Логин пользователя	шифр	название	автор	жанр	тип	издательство	год выпуска	статус	Выбрать
763ce56d-6f96-4add-953f-b06e55243ea	1e9e66e0-a2ba-44f9-826c-9b257f042b58	1	string	string	string, Zxc Sda AAw	string	string	MyFirstPublisher	2007	Не возвращено	<input checked="" type="checkbox"/>
e62a3a79-6615-4060-8154-42313d4b0eee	a7086f39-f8c1-4920-855d-4d169cef131c	1	string	string	string, Zxc Sda AAw	string	string	MyFirstPublisher	2005	Возвращено	<input type="checkbox"/>
b3299ff6-2582-496b-bfda-f58499388e87	01025405-9f24-48b1-abac-2a44540ed455	1	string	string	string, Zxc Sda AAw	string	string	MyFirstPublisher	2015	Не возвращено	<input checked="" type="checkbox"/>
ef032c3d-2213-4b24-8961-8e3f4903bcb0	b71c32d3-cbab-46a0-8993-d7bce21d008c	1	string	string	string, Zxc Sda AAw	string	string	MyFirstPublisher	2012	Возвращено	<input type="checkbox"/>

Рисунок 2.23 Интерфейс админа для отметки возврата книг(и)

На рисунке продемонстрирован интерфейс выделения админом книг, а также представлен интерфейс кнопок выполнения кода по нажатию на них.

Имеющийся функционал:

- 1) Кнопка подтверждения возврата;
- 2) Возможность выбора нескольких рядов из таблицы книг.



guid заказа	guid экземпляра	шифр	название	автор	жанр	тип	издательство	год выпуска	статус
52dde908-7831-4ee3-941e-e8837554aba4	b71c32d3-cbab-46a0-8993-d7bce21d008c	string	string	string, Zxc Sda AAaw	string	string	MyFirstPublisher	2012	Возвращено
4ebceafa-cc6f-44d5-b38c-aa9198cb803e	a97bbee3-7fe3-40bb-a190-477c27ed33bd	string	string	string, Zxc Sda AAaw	string	string	MyFirstPublisher	2010	Возвращено
8aec2adb-8271-43b1-a198-e511c75a5d37	a7086f39-f8c1-4920-855d-4d169cef131c	string	string	string, Zxc Sda AAaw	string	string	MyFirstPublisher	2005	Не возвращено
012deea2-fb6e-43e8-8bc1-c2200905daca	00ccf050-28fd-4033-bc5e-093b11880431	string	string	string, Zxc Sda AAaw	string	string	MyFirstPublisher	2020	Не возвращено

Рисунок 2.24 Интерфейс пользователя с информацией о книгах.

На рисунке продемонстрирован интерфейс информации пользователя о его взятых книгах, а также о возвращенных им книгах за все время. Сам пользователь не имеет кнопки подтверждения возврата книги, ее имеют только администраторы. Из-за чего в этом пользовательском меню нет кнопок, оно нужно для осведомления пользователя о его взаимодействии с книгами.

```
public Guid? CreateVacancy(Guid uIdUser, string text)
{
    var user = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.User>().SingleOrDefault(x => x.UidUser == uIdUser);
    if (user == null) return null;
    var vacancy = new DatabaseAccessLayer.Entities.Vacancy
    {
        UidVacancy = Guid.NewGuid(),
        Text = text,
        StatusOfVacancy = 0,
        User = user
    };
    _libraryWebDbContext.Add(vacancy);
    _libraryWebDbContext.SaveChanges();
    return vacancy.UidVacancy;
}
```

Ссылка 1

Рисунок 2.25 Код отправки вакансии.

На рисунке продемонстрирован код отправки пользователем вакансии. Происходит подключение к базе данных, после чего вызывается функция добавления вакансии в таблицу вакансий.

Создать вакансию

fdsfsdfsdf

Создать вакансию

Мои вакансии

guid вакансии	Текст	Статус вакансии
a8ddb3df-17c6-4feb-95ea-c19a3eba922c	asdasdasdasdasd	Отказано
1b792e94-bd4a-43c3-b3f1-08b823767543	dadasd	На рассмотрении
36414e0d-6985-4412-8c91-9039497f9934	asdasd	На рассмотрении
5283446d-ac61-44a5-a8b8-36ed4347e9bb	dddd	На рассмотрении
a939602b-b0a6-47bb-9399-5819307cf639	asasdasd	На рассмотрении
d46a05eb-5b1d-4b1e-bf6d-defd00731c08	hghfghfghfghfgh	На рассмотрении
2335a8c8-d1b9-4fb7-900c-1bb1c07ad967	fghfghfghfgh	На рассмотрении
984cfc09-4182-4851-89ce-bb6dec45b676	dddd	На рассмотрении

Рисунок 2.26 Интерфейс отправки вакансии

На рисунке продемонстрирован пользовательский интерфейс вкладки «Вакансии». Каждый пользователь может написать вакансию, после чего нажать на кнопку «Отправить» для ее отправки. История всех его вакансий, а также их статусах содержится в таблице «Мои вакансии» и доступна каждому пользователю.

```
public Guid? EditData(Guid uIdUser, UserUpdate userUpdate)
{
    var user = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.User>().SingleOrDefault(x => x.UidUser == uIdUser);
    if (user == null) return null;
    if ((_libraryWebDbContext.Set<DatabaseAccessLayer.Entities.User>().SingleOrDefault(x => x.Login == userUpdate.Login) != null) && (userUpdate.Login != user.Login)
        || (_libraryWebDbContext.Set<DatabaseAccessLayer.Entities.User>().SingleOrDefault(x => x.Email == userUpdate.Email) != null) && (userUpdate.Email != user.Email))
        return null;
    user.Name = userUpdate.Name;
    user.Surname = userUpdate.Surname;
    user.Otchestvo = userUpdate.Otchestvo;
    user.Login = userUpdate.Login;
    user.Email = userUpdate.Email;
    user.Password = GetHash(userUpdate.Password);
    user.Address = userUpdate.Address;
    user.PhoneNumber = userUpdate.PhoneNumber;
    user.Age = userUpdate.Age;
    _libraryWebDbContext.SaveChanges();
    return user.UidUser;
}
```

Рисунок 2.27 Код изменения данных пользователя.

Этот код демонстрирует подключение к базе данных. После чего вызывается функция изменения данных пользователя, которые он ввел во вкладке «Профиль»

The screenshot displays a web interface for user profile management, divided into two main sections: 'Данные пользователя' (User Data) and 'Изменить данные' (Change Data).

Данные пользователя (User Data): This section lists the current user information: Email: 2@mail.ru, Имя: 22, Фамилия: 22, Отчество: 22, Логин: 2, Адрес: 22, Номер телефона: 22, and Возраст: 22. Below this list is a button labeled 'Выйти из аккаунта' (Logout).

Изменить данные (Change Data): This section contains a form with input fields for each field: Email, Имя, Фамилия, Отчество, Логин, Пароль, Адрес, Номер телефона, and Возраст. A 'Сохранить' (Save) button is located at the bottom of the form.

Рисунок 2.28 Интерфейс изменения данных пользователя.

На рисунке продемонстрирован пользовательский интерфейс вкладки «Профиль». Каждый пользователь может изменить хранящиеся о нем данные, путем ввода их в текстовые строки. Если проверки на заполнение обязательных полей и на занятость нового логина успешно выполнены, то после нажатия на кнопку «Сохранить», вся новая информация о пользователе будет изменена.


```

public Guid? CreateBook(BookUpdate bookUpdate)
{
    var book = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Book>()
        .SingleOrDefault(x => x.NameOfBook == bookUpdate.NameOfBook
        && x.Shifr == bookUpdate.Shifr);
    if (book != null) return null;

    book = new DatabaseAccessLayer.Entities.Book
    {
        UidBook = Guid.NewGuid(),
        NameOfBook = bookUpdate.NameOfBook,
        Shifr = bookUpdate.Shifr,
    };

    var authors = new List<DatabaseAccessLayer.Entities.Author>();
    foreach (var authorOfBook in bookUpdate.AuthorsOfBook)
    {
        var author = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Author>()
            .SingleOrDefault(x => x.NameOfAuthor == authorOfBook);
        if (author == null)
        {
            author = new DatabaseAccessLayer.Entities.Author
            {
                UidAuthor = Guid.NewGuid(),
                NameOfAuthor = authorOfBook
            };
            _libraryWebDbContext.Add(author);
        }
        authors.Add(author);
    }

    var genres = new List<DatabaseAccessLayer.Entities.Genre>();
    foreach (var genreOfBook in bookUpdate.GenresOfBook)
    {
        var genre = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Genre>()
            .SingleOrDefault(x => x.NameOfGenre == genreOfBook);
        if (genre == null)
        {
            genre = new DatabaseAccessLayer.Entities.Genre
            {
                UidGenre = Guid.NewGuid(),
                NameOfGenre = genreOfBook
            };
            _libraryWebDbContext.Add(genre);
        }
        genres.Add(genre);
    }

    var typesBk = new List<DatabaseAccessLayer.Entities.TypeBk>();
    foreach (var typeBkOfBook in bookUpdate.TypesBkOfBook)
    {
        var typeBk = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.TypeBk>()
            .SingleOrDefault(x => x.NameOfTypeBk == typeBkOfBook);
        if (typeBk == null)
        {
            typeBk = new DatabaseAccessLayer.Entities.TypeBk
            {
                UidTypeBk = Guid.NewGuid(),
                NameOfTypeBk = typeBkOfBook
            };
            _libraryWebDbContext.Add(typeBk);
        }
    }
}

```



```

public bool DeleteBook(Guid uIdBook)
{
    var book = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Book>()
        .Include(x => x.AuthorsOfBook)
        .Include(x => x.GenresOfBook)
        .Include(x => x.TypesBkOfBook)
        .Include(x => x.ExamplesWithBook)
        .SingleOrDefault(x => x.UidBook == uIdBook);
    if (book == null) return false;
    try
    {
        _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Book>().Remove(book);
        _libraryWebDbContext.SaveChanges();
    }
    catch { return false; }
    return true;
}

```

Рисунок 2.29 Код изменения данных о книге/добавление книги.

Этот код демонстрирует подключение к базе данных. После чего проходит по всем выбранным админом полям и делает проверку. Если айди экземпляра выбранного ряда пусто, то значит экземпляр еще не добавлен, а значит нужно вызвать функцию добавления нового экземпляра. Если же поле индекса не пусто, то вызывается функция редактирования книги.

The screenshot shows a web application interface for managing a library database. It includes several sections with tables and action buttons:

- Авторы (Authors):** A table with columns: Guid, Автор, and Выбрать. It contains 6 rows of author data.
- Жанры (Genres):** A table with columns: Guid, Жанр, and Выбрать. It contains 6 rows of genre data.
- Типы (Types):** A table with columns: Guid, Тип, and Выбрать. It contains 4 rows of type data.
- Книги (Books):** A table with columns: Шифр, Название, Авторы, Жанры, Типы, and Выбрать. It contains 6 rows of book data.
- Издательства (Publishers):** A table with columns: Guid, Название, and Выбрать. It contains 5 rows of publisher data.
- Все экземпляры (All Examples):** A table with columns: Год выпуска, Guid, Шифр, Название, Автор, Жанр, Тип, Издательство, Год выпуска, Статус, and Выбрать. It contains 6 rows of example data.

Each table has buttons for 'Добавить' (Add) and 'Удалить выбранные' (Delete selected) at the top. The 'Выбрать' (Select) column in each table contains checkboxes for row selection.

Рисунок 2.30 Интерфейс изменения данных о книге/добавление книги

На рисунке продемонстрирован интерфейс администратора вкладки «Изменить книги». Каждый администратор может изменить хранящиеся о

каждом экземпляре данные, путем изменения информации в колонке. Либо же в последнюю пустую строку ввести новые данные. Тогда при нажатии на кнопку «Сохранить». Все выделенные ряды проверятся на изменения, и если изменения будут обнаружены, то они сохранятся для этой книги. Если же книги до этого не существовало, то такой экземпляр добавится, а все его существующие атрибуты присвоятся ему. В случае, если, к примеру, введенного жанра новой книги до этого не существовало, то он создастся и присвоится новой книге. При нажатии на кнопку «Удалить экземпляр», удаляется экземпляр из таблицы Example, а также все заказы с этим экземпляром в таблице Order будут удалены.

```
public bool DeleteExample(Guid uIdExample)
{
    var example = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Example>()
        .Include(x => x.Book)
        .Include(x => x.Publisher)
        .SingleOrDefault(x => x.UidExample == uIdExample);
    var order = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Order>()
        .Include(x => x.Example)
        .Where(x => x.Example.UidExample == uIdExample)
        .ToList();
    if (example == null) return false;
    try
    {
        foreach(var el in order)
        {
            _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Order>().Remove(el);
        }
        _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Example>().Remove(example);
        _libraryWebDbContext.SaveChanges();
    }
    catch { return false; }
    return true;
}
```

Рисунок 2.31 Код удаления экземпляра

Этот код демонстрирует подключение к базе данных. После чего вызывается процедура, которая удаляет экземпляр и заказы с ним.

```

Ссылка: 1
public bool AcceptVacancy(Guid uIdVacancy)
{
    var vacancy = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Vacancy>()
        .Include(x => x.User)
        .SingleOrDefault(x => x.UidVacancy == uIdVacancy);
    if (vacancy == null) return false;
    vacancy.StatusOfVacancy = 1;
    vacancy.User.IsAdmin = true;
    _libraryWebDbContext.SaveChanges();
    return true;
}

Ссылка: 1
public bool RejectVacancy(Guid uIdVacancy)
{
    var vacancy = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Vacancy>()
        .Include(x => x.User)
        .SingleOrDefault(x => x.UidVacancy == uIdVacancy);
    if (vacancy == null) return false;
    vacancy.StatusOfVacancy = -1;
    vacancy.User.IsAdmin = false;
    _libraryWebDbContext.SaveChanges();
    return true;
}

Ссылка: 1
public bool DeleteVacancy(Guid uIdVacancy)
{
    var vacancy = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Vacancy>()
        .Include(x => x.User)
        .SingleOrDefault(x => x.UidVacancy == uIdVacancy);
    if (vacancy == null) return false;
    _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Vacancy>().Remove(vacancy);
    _libraryWebDbContext.SaveChanges();
    return true;
}

Ссылка: 1
public bool DeleteVacancy(Guid uIdVacancy, Guid uIdUser)
{
    var vacancy = _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Vacancy>()
        .Include(x => x.User)
        .SingleOrDefault(x => x.UidVacancy == uIdVacancy);
    if (vacancy == null || vacancy.User.UidUser != uIdUser) return false;
    try
    {
        _libraryWebDbContext.Set<DatabaseAccessLayer.Entities.Vacancy>().Remove(vacancy);
        _libraryWebDbContext.SaveChanges();
    }
    catch { return false; }
    return true;
}

```

Рисунок 2.32 Код отклонения или принятия заявки пользователя.

Этот код демонстрирует подключение к базе данных и вызова функции отмены или принятия вакансии (в зависимости от нажатой кнопки). Если вакансию приняли, то создателю вакансии присваивается статус админа.

Все вакансии

Принять вакансии	Отклонить вакансии					
Guid вакансии	Логин пользователя	Почта пользователя	Текст	Статус вакансии	Выбрать	
52848abc-b2df-488d-bc28-c69fa07b38c3	3	3@mail.ru	Wafasefsdf	Отказано	<input type="checkbox"/>	
dfa7fa9a-2bfb-41e1-8ac1-63580794c583	3	3@mail.ru	asdasd	Отказано	<input type="checkbox"/>	
fd0544f6-6f16-4789-b19c-e07888790be4	3	3@mail.ru	Я ХОЧУ РАБОТАТЬ	Отказано	<input type="checkbox"/>	
93a36a05-4f57-4206-848e-94fb75d14ea5	3	3@mail.ru	Я ХОЧУ РАБОТАТЬ !!!!!!!	Отказано	<input type="checkbox"/>	
a8ddb3df-17c6-4feb-95ea-c19a3eba922c	2	2@mail.ru	asdasdasdasdasd	Отказано	<input type="checkbox"/>	
1b792e94-bd4a-43c3-b3f1-08b823767543	2	2@mail.ru	dadasd	На рассмотрении	<input checked="" type="checkbox"/>	
36414e0d-6985-4412-8c91-9039497f9934	2	2@mail.ru	asdasd	На рассмотрении	<input checked="" type="checkbox"/>	
5283446d-ac61-44a5-a8b8-36ed4347e9bb	2	2@mail.ru	dddd	На рассмотрении	<input type="checkbox"/>	
a939602b-b0a6-47bb-9399-5819307cf639	2	2@mail.ru	asdasd	На рассмотрении	<input type="checkbox"/>	
d46a05eb-5b1d-4b1e-bf6d-defd00731c08	2	2@mail.ru	hghfghfghfghfgh	На рассмотрении	<input checked="" type="checkbox"/>	
2335a8c8-d1b9-4fb7-900c-1bb1c07ad967	2	2@mail.ru	fghfghfghfgh	На рассмотрении	<input type="checkbox"/>	
984cfc09-4182-4851-89ce-bb6dec45b676	2	2@mail.ru	dddd	На рассмотрении	<input type="checkbox"/>	

Рисунок 2.33 Интерфейс отклонения или принятия заявки пользователя

На рисунке продемонстрирован интерфейс администратора вкладки «Вакансии». Каждый администратор может видеть всю информацию о активных вакансиях пользователей и принимать их, либо отклонять. Все рассмотренные вакансии будут перемещены в таблицу «История вакансий» с их новыми статусами: принято или отклонено. Если вакансия была принята, то пользователю, отправившему эту вакансию, присвоится статус администратора.

Использование контрактов также помогает упростить процесс разработки и поддержки системы. За счет четкого определения интерфейсов и требований к взаимодействию, разработчики могут работать над отдельными компонентами системы независимо друг от друга. Кроме того, любые изменения в контракте должны быть явно разрешены и координированы между всеми заинтересованными сторонами, что помогает предотвратить непредвиденные и несовместимые изменения в системе.

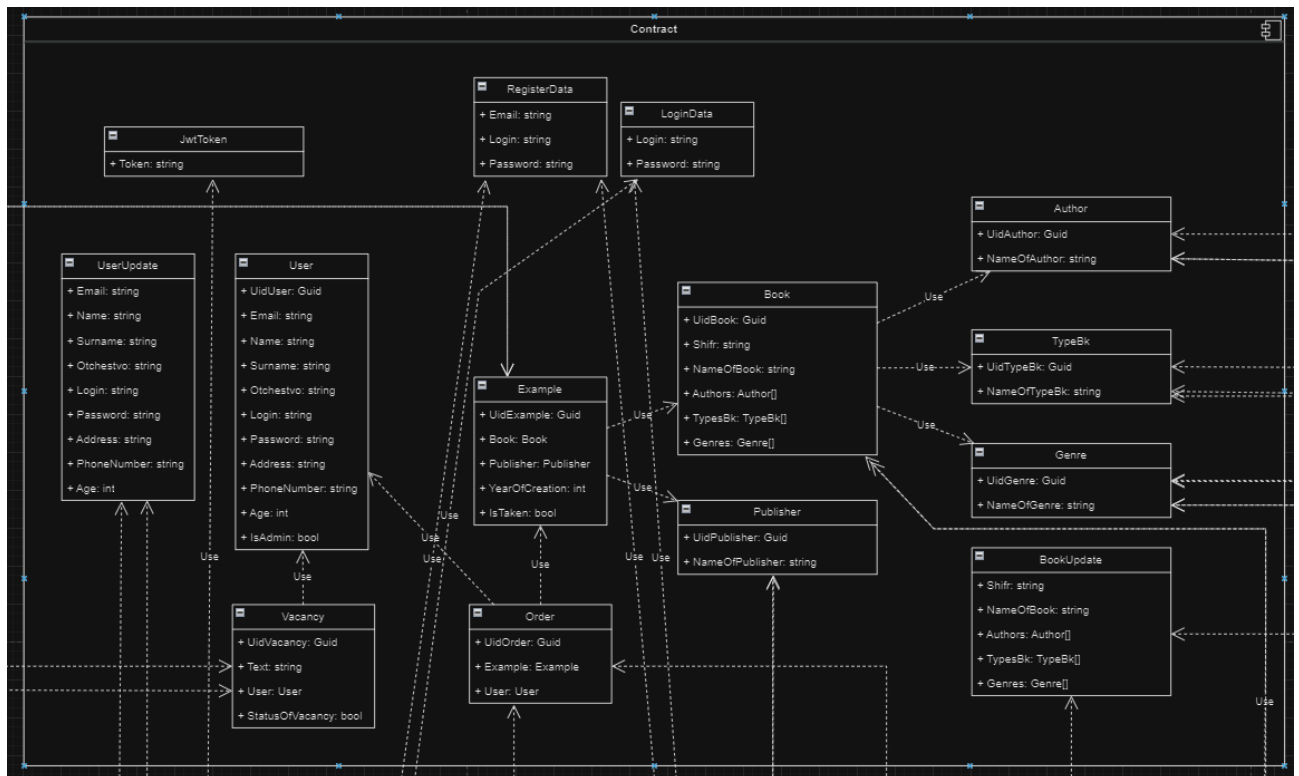


Рисунок 2.33 Contracts

API также позволяет обеспечить высокую степень гибкости и переиспользования кода, так как разработчики могут использовать предоставленные функции и методы для создания своих собственных приложений и сервисов. Помимо этого, использование API позволяет упростить процесс разработки и улучшить уровень безопасности, так как внешние компоненты могут обращаться к системе только через определенные интерфейсы, защищенные от нежелательного доступа и изменений.

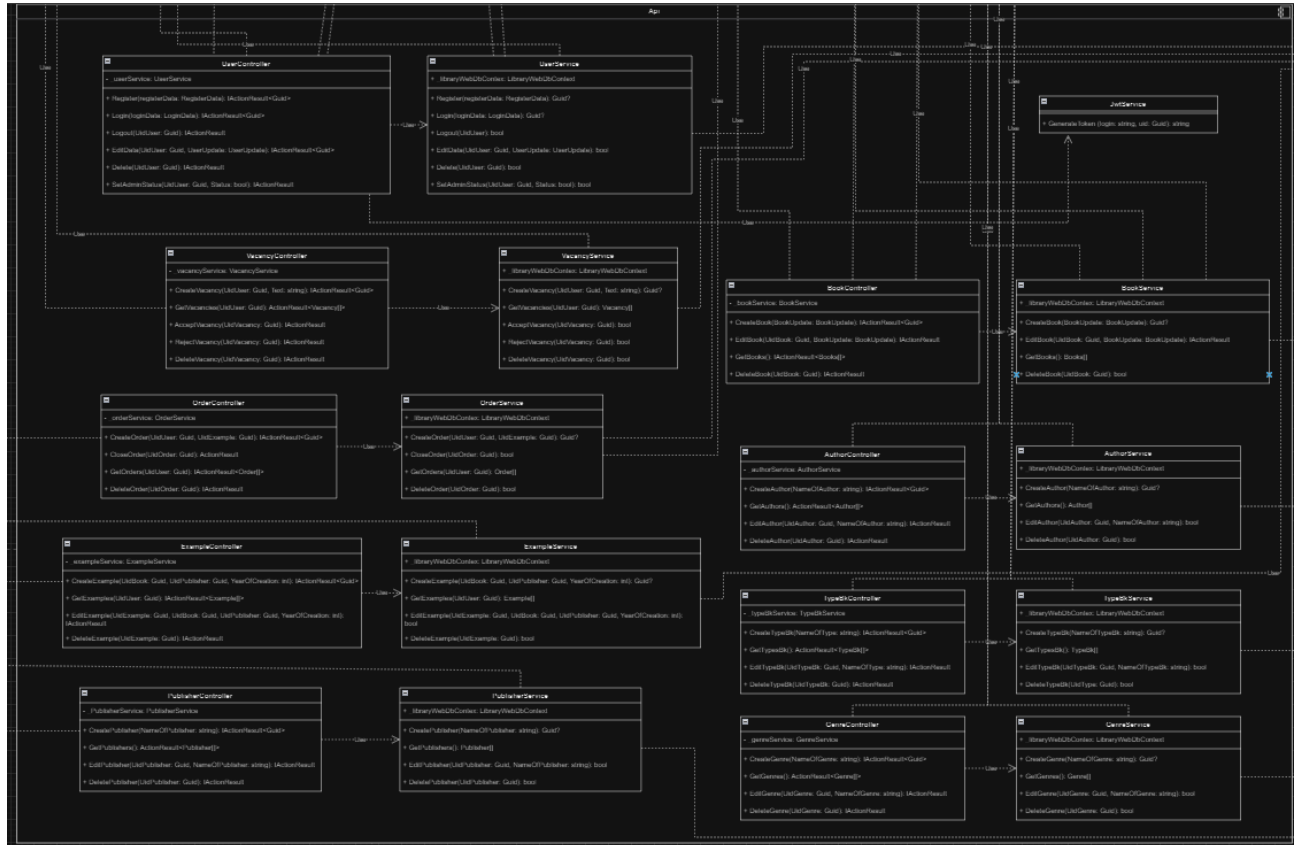


Рисунок 2.34 API

Database Access Layer (DAL) - это компонент в backend разработке, который служит для управления доступом и взаимодействием с базой данных. DAL абстрагирует работу с базой данных от остальной части приложения, предоставляя удобные и единообразные методы для выполнения операций чтения, записи и обновления данных.

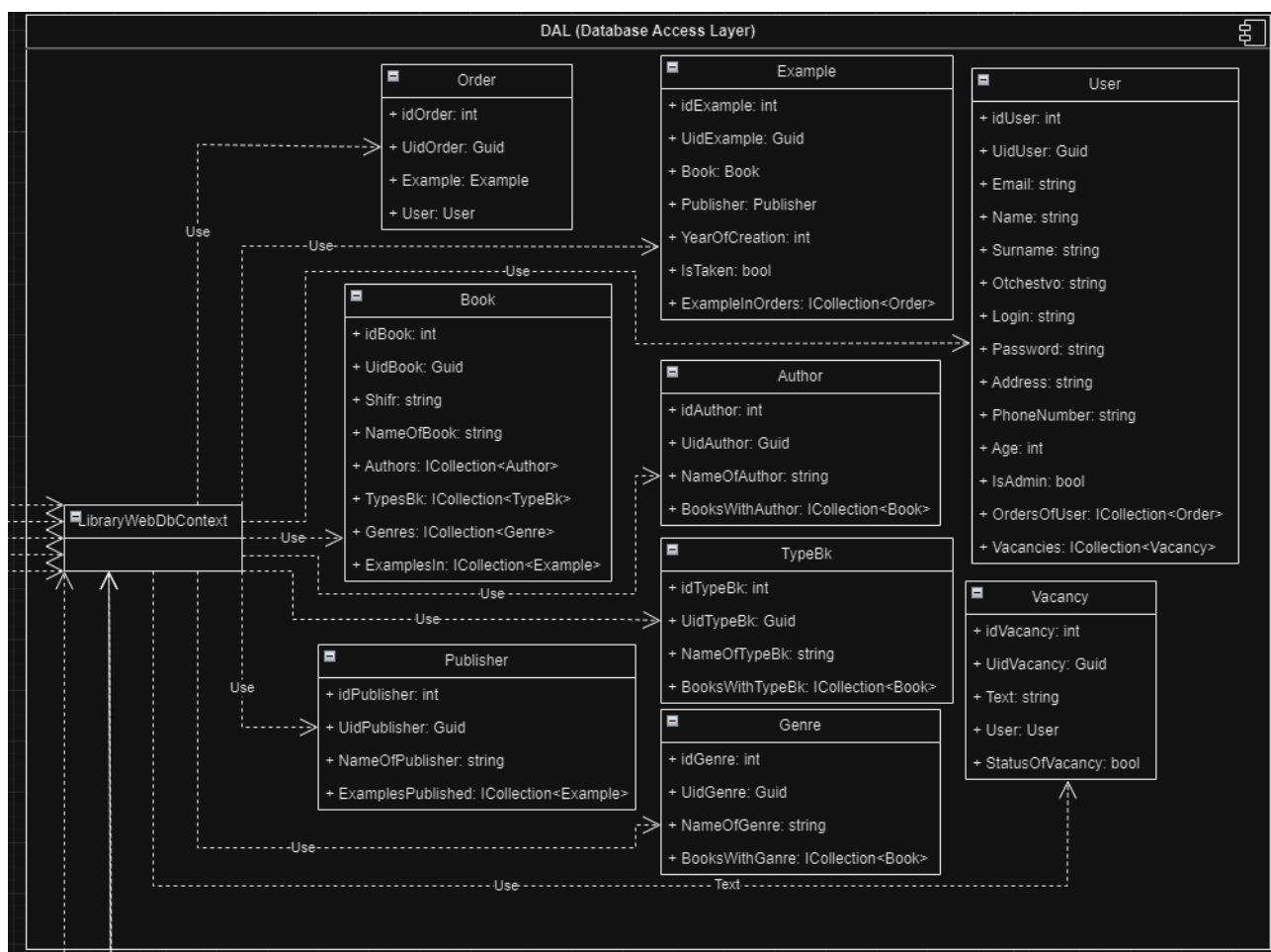


Рисунок 2.35 API

ЗАКЛЮЧЕНИЕ

В рамках данного проекта была разработана информационная система YourLibrary, которая обеспечивает процессы аренды и возврата книг в библиотеке. Для ее реализации были выбраны наиболее эффективные инструменты, такие как Microsoft Visual Studio и язык программирования C#, а также система управления базами данных MSSQL. Благодаря использованию таких инструментов, удалось создать высокопроизводительную и функциональную информационную систему, которая упрощает работу библиотеки и повышает ее эффективность. В процессе реализации проекта были получены ценные навыки в проектировании баз данных, создании десктопного приложения и подключении базы данных к приложению. В итоге была разработана информационная система, которая значительно упрощает процессы аренды и возврата книг в библиотеке и повышает качество ее работы.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

- 1) К. Дж. Дейт «Введение в системы баз данных»
- 2) Рамез Эльмасри, Шамкант Б. Навате «Основы систем баз данных»
- 3) Абрахам Сильбершац, Генри Корт, Сударшан С. «Концепции систем баз данных»
- 4) Гектор Гарсия-Молина, Джеффри Ульман, Дженнифер Уидом «Системы баз данных: Полная книга»
- 5) Кристофер Дж. Дейт «Проектирование баз данных и реляционная теория: Нормальные формы и все, что с этим связано»
- 6) Джо Селко «SQL для умников: Продвинутое программирование SQL»
- 7) Мартин Фаулер «Шаблоны архитектуры корпоративных приложений»
- 8) Майкл Хернандес «Проектирование баз данных для профанов: Практическое руководство по проектированию реляционных баз данных»
- 9) Вадим Огиевецкий «SQL Performance Explained»
- 10) Крэйг Маллинс «Руководство разработчика DB2»
- 11) Курс баз данных для 3 и 4 семестров по направлению 09.03.02 «Информационные системы и технологии»