Министерство образования и науки Кыргызской Республики

Кыргызский Государственный Технический Университет

им. И. Раззакова

Кафедра программного обеспечения компьютерных систем

БАКАЛАВРСКАЯ УЧЕБНАЯ ПРОГРАММА

НАПРАВЛЕНИЕ—710400 «ПРОГРАММНАЯ ИНЖЕНЕРИЯ»

Дисциплина «Учебная практика»

**ОТЧЕТ**

**ПО ВЫПОЛНЕНОЙ РАБОТЕ**

по дисциплине

«УЧЕБНАЯ ПРАКТИКА»

Выполнил: студент группы ПИ-1-15

Аюпов А.

Проверил: кандидат технических наук, профессор

Тен Иосиф Григорьевич

СОДЕРЖАНИЕ

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ СИСТЕМЫ ДЛЯ СТУДЕНЧЕСКОЙ БИБЛИОТЕКИ 1](#_Toc524297194)

[1. Введение 1](#_Toc524297195)

[1.1. Аббревиатуры и обозначения 1](#_Toc524297196)

[1.2. Назначение 1](#_Toc524297197)

[1.3. Предмет 1](#_Toc524297198)

[1.4. Цель создания 1](#_Toc524297199)

[1.5. Практичность 2](#_Toc524297200)

[2. Бизнес-требования 2](#_Toc524297201)

[3. Функциональные требования 2](#_Toc524297202)

[3.1. Пользовательские требования 2](#_Toc524297203)

[3.2. Категории пользователей 2](#_Toc524297204)

[4. Нефункциональные требования 2](#_Toc524297205)

[4.1. Производительность 2](#_Toc524297206)

[4.2. Безопасность 2](#_Toc524297207)

[4.3. Требования к данным 2](#_Toc524297208)

[4.4. Требования к преобразованию данных 2](#_Toc524297209)

[4.5. Требования к пользовательской документации 3](#_Toc524297210)

[4.6. Требования к лицензированию 3](#_Toc524297211)

[5. Функциональное описание решения 3](#_Toc524297212)

[5.1. Интерфейсы 3](#_Toc524297213)

[5.5. Источники данных 3](#_Toc524297214)

[5.6. Преобразование данных 3](#_Toc524297215)

[5.7. Обработка ошибок 4](#_Toc524297216)

[5.8. Проектные требования и ограничения 4](#_Toc524297217)

[6. Дополнение 4](#_Toc524297218)

[IDEF0-Диаграмма (AS IS) 5](#_Toc524297219)

[Реализация 6](#_Toc524297220)

[1. Структура проекта 6](#_Toc524297221)

[2. Листинг основных классов 7](#_Toc524297222)

[3. Скриншоты 11](#_Toc524297223)

[Тестирование 14](#_Toc524297224)

[Заключение 16](#_Toc524297225)

[Источники 16](#_Toc524297226)

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ СИСТЕМЫ ДЛЯ СТУДЕНЧЕСКОЙ БИБЛИОТЕКИ

## Введение

### Аббревиатуры и обозначения

ASP (Active Service Page) – Фреймворк.

MVC (Model View Controller) – Паттерн.

N-Layer Архитектура – (Н-Слой) архитектура, в которой приложение разбивается на три логических уровня.

DAL (Data Access Layer) – уровень доступа к данным.

BLL (Business Logic Layer) – уровень бизнес логики.

WEB – здесь, уровень представления.

ПП – Программный Продукт.

ПО – Программное Обеспечение.

### Назначение

Данный документ содержит описание проекта, его предметную область и характеристики. Также содержит всю необходимую информацию о сроках и ответственных лицах, участвующих в разработке ПО.

### Предмет

Предметом данного документа является описание требований к web-ориентированной системе для библиотекарей, а также её особенности.

### Цель создания

Описание проблемы

На данный момент бизнес-процессы, происходящие в библиотеке не автоматизированы. Библиотекарям приходится хранить всю информацию о книгах и студентах в бумажной форме в больших стеллажах, порой выделяя для этих целей отдельную комнату. При единовременном приёме/выдачи большого количества книг создаётся коллапс, в виду невозможности ускорения процесса.

Целью создания данного ПП является автоматизация деятельности библиотекаря на стадии записи новых студентов, книг, а также выдачи и приёма книг. Данное приложение позволит «в пару кликов» получать и выдавать книги, а также исключит необходимость в хранении карточек в физическом виде.

### Практичность

Система позволяет избежать бумажной волокиты, а также исключает возможность потери соответствующих бумаг.

## Бизнес-требования

Повысить производительность библиотеки за счёт автоматизирования процессов приёма, хранения и выдачи книг. Сократить время, затрачиваемое библиотекарем на них.

## Функциональные требования

### Пользовательские требования

* Возможность добавления новых студентов.
* Возможность добавления новых групп.
* Возможность добавления новых книг.
* Возможность выдачи необходимого количества книг студенту.
* Возможность приёма книг от студента.
* Возможность удаления студентов.

### Категории пользователей

**Admin** – библиотекарь. Доступен весь функционал системы.

## Нефункциональные требования

### Производительность

Обработка запросов не должна длиться дольше минуты.

### Безопасность

Так как система не имеет выхода в интернет, требований к безопасности не предъявлялось.

### Требования к данным

Требований нет.

### Требования к преобразованию данных

Требований нет.

### Требования к пользовательской документации

Требуется наличие полной и подробной документации по всем функциям, включающую в себя скриншоты.

### Требования к лицензированию

Не обязательно.

## Функциональное описание решения

#### Интерфейсы

#### Пользовательский интерфейс

Интерфейс должен быть интуитивно понятен и представлять из себя сайт.

#### Аппаратный интерфейс

##### NET Framework

Платформа Microsoft .NET Framework версии 4.0 или выше.  
**Для работы сайта обязателен режим работы пула приложения ASP.NET - integrated.**

##### Дисковое пространство

Система занимает 500 Mб под хранение исполняемых файлов и требует минимум 1 Tб под базу данных.

#### Программный интерфейс

Cервер должен работать под управлением одной из следующих операционных систем:  
     - Windows Server 2012 / 2012 R2  
     - Windows Server 2008 / 2008 R2

В системе должен быть установлен компонент IIS (Internet Information Services) версии 7 и выше.

Доступ к веб-сайту осуществляется с помощью одного из современных веб-браузеров (Opera, Internet Explorer, Google Chrome и прочее).

### Источники данных

Основным источником данных является база данных, которая содержит информацию о студентах, книгах, группах и заёмах книг определенными студентами.

### Преобразование данных

Все текстовые данные преобразуются в кодировке UTF-8.

### Обработка ошибок

Основные ошибки:

1. Ошибка 404 – файл не найден.
2. Ошибка 500 – внутренняя ошибка сервера.

Описанные выше ошибки обрабатываются системой и выдаются в представление через веб-интерфейс.

### Проектные требования и ограничения

Система должна реализовывать многоуровневую архитектуру MVC-паттерна. Системой управления базой данных является SQL Server. Используются язык разметки HTML, каскадные таблицы стилей CSS, язык программирования C# с использованием JavaScript.

## Дополнение

В случае обнаружении необходимости разработчик имеет право вносить корректировки.

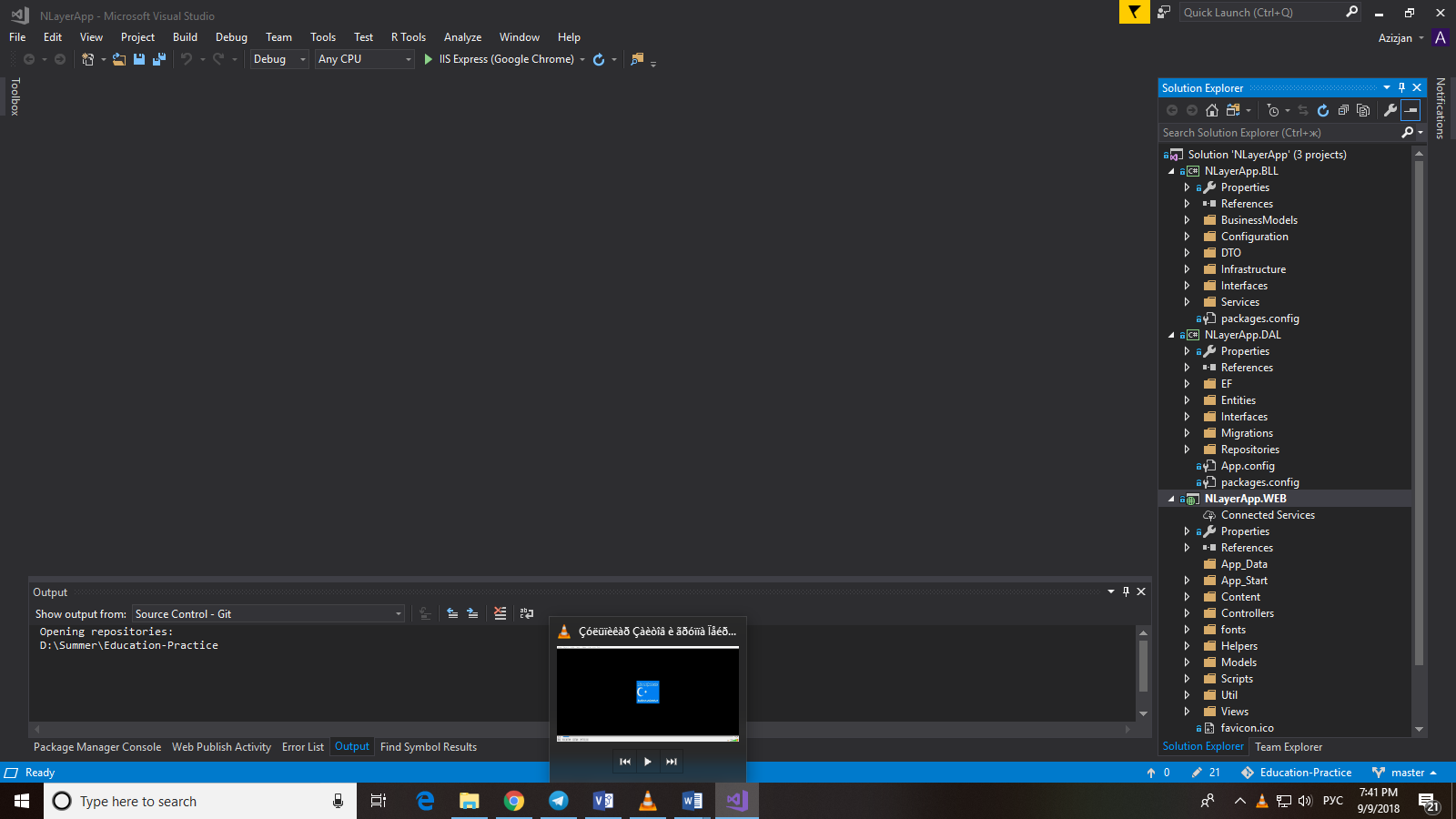
# IDEF0-Диаграмма (AS IS)





# Реализация

## Структура проекта



## Листинг основных классов

public class BookLeaseDTO

{

public int LeaseId { get; set; }

public int StudentId { get; set; }

public StudentDTO Student { get; set; }

public int BookId { get; set; }

public BookDTO Book { get; set; }

public int Amount { get; set; }

public DateTime GetTime { get; set; }

public DateTime? ReturnTime { get; set; }

}

public class LeaseService : ILeaseService

{

IUnitOfWork Database { get; set; }

public LeaseService(IUnitOfWork uow)

{

Database = uow;

}

public bool MakeLease(BookLeaseDTO bookLeaseDTO)

{

Book book = Database.Books.Get(bookLeaseDTO.BookId);

if (book == null)

throw new ValidationException("The book was not found", "");

if (book.Amount >= bookLeaseDTO.Amount)

{

BookLease bookLease = new BookLease

{

Amount = bookLeaseDTO.Amount,

StudentId = bookLeaseDTO.StudentId,

BookId = bookLeaseDTO.BookId,

GetTime = bookLeaseDTO.GetTime,

ReturnTime = bookLeaseDTO.ReturnTime

};

book.Amount -= bookLeaseDTO.Amount;

Database.Books.Update(book);

Database.Save();

Database.BookLeases.Create(bookLease);

Database.Save();

return true;

}

else

return false;

}

public IEnumerable<BookLeaseDTO> GetLeases()

{

            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<BookLease, BookLeaseDTO>()).CreateMapper();

return mapper.Map<IEnumerable<BookLease>, List<BookLeaseDTO>>(Database.BookLeases.GetAll());

}

public IEnumerable<BookDTO> GetBooks()

{

            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<Book, BookDTO>()).CreateMapper();

return mapper.Map<IEnumerable<Book>, List<BookDTO>>(Database.Books.GetAll());

}

public IEnumerable<GroupDTO> GetGroups()

{

            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<Group, GroupDTO>()).CreateMapper();

return mapper.Map<IEnumerable<Group>, List<GroupDTO>>(Database.Groups.GetAll());

}

public IEnumerable<BookLeaseDTO> GetBookLeases()

{

            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<BookLease, BookLeaseDTO>()).CreateMapper();

return mapper.Map<IEnumerable<BookLease>, List<BookLeaseDTO>>(Database.BookLeases.GetAll());

}

public void Return(int? id)

{

if (id == null)

throw new ValidationException("Lease ID is not set", "");

var lease = Database.BookLeases.Get(id.Value);

if (lease == null)

throw new ValidationException("The lease was not found", "");

lease.ReturnTime = DateTime.Now;

Database.BookLeases.Update(lease);

Database.Save();

var book = Database.Books.Get(lease.BookId);

book.Amount += lease.Amount;

Database.Books.Update(book);

Database.Save();

}

public void Dispose()

{

Database.Dispose();

}

public StudentDTO GetStudent(int? id)

{

if (id == null)

throw new ValidationException("Student ID is not set", "");

var student = Database.Students.Get(id.Value);

if (student == null)

throw new ValidationException("The student was not found", "");

return new StudentDTO { StudentId = student.StudentId, FirstName = student.FirstName, LastName = student.LastName, GroupId = student.GroupId };

}

}

public class BookLeaseRepository : IRepository<BookLease>

{

private MainContext db;

public BookLeaseRepository(MainContext context)

{

this.db = context;

}

public IEnumerable<BookLease> GetAll()

{

return db.BookLeases;

}

public BookLease Get(int id)

{

return db.BookLeases.Find(id);

}

public void Create(BookLease bookLease)

{

db.BookLeases.Add(bookLease);

}

public void Update(BookLease bookLease)

{

db.Entry(bookLease).State = EntityState.Modified;

}

public IEnumerable<BookLease> Find(Func<BookLease, Boolean> predicate)

{

return db.BookLeases.Where(predicate).ToList();

}

public void Delete(int id)

{

BookLease bookLease = db.BookLeases.Find(id);

if (bookLease != null)

db.BookLeases.Remove(bookLease);

}

}

public class LeasesController : Controller

{

ILeaseService leasesService;

IStudentsService studentsService;

IBooksService booksService;

public LeasesController(ILeaseService serv, IStudentsService studServ, IBooksService bookServ)

{

leasesService = serv;

studentsService = studServ;

booksService = bookServ;

}

// GET: Return

public ActionResult Return(int? Id)

{

if (Id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

leasesService.Return(Id);

return RedirectToAction("Index");

}

// GET: Leases

public ActionResult Lease(int? Id)

{

BookDTO bookDTO = booksService.GetBook(Id);

if(bookDTO != null)

{

BookLeaseVM bookLeaseVM = new BookLeaseVM

{

BookId = bookDTO.BookId,

Name = bookDTO.Name,

Author = bookDTO.Author

};

ViewBag.StudentId = new SelectList(studentsService.GetStudents(), "StudentId", "FirstName");

return View(bookLeaseVM);

}

return View();

}

[HttpPost]

public ActionResult Lease(BookLeaseVM bookLeaseVM)

{

try

{

BookLeaseDTO bookLeaseDTO = new BookLeaseDTO

{

BookId = bookLeaseVM.BookId,

StudentId = bookLeaseVM.StudentId,

GetTime = DateTime.Now,

Amount = bookLeaseVM.Amount

};

if(leasesService.MakeLease(bookLeaseDTO))

return RedirectToAction("Index");

return View("Error");

}

catch

{

return View();

}

}

public ActionResult Index()

{

IEnumerable<BookLeaseDTO> leaseDtos = leasesService.GetLeases();

var leases = Mapper.Map<List<LeaseListVM>>(leaseDtos);

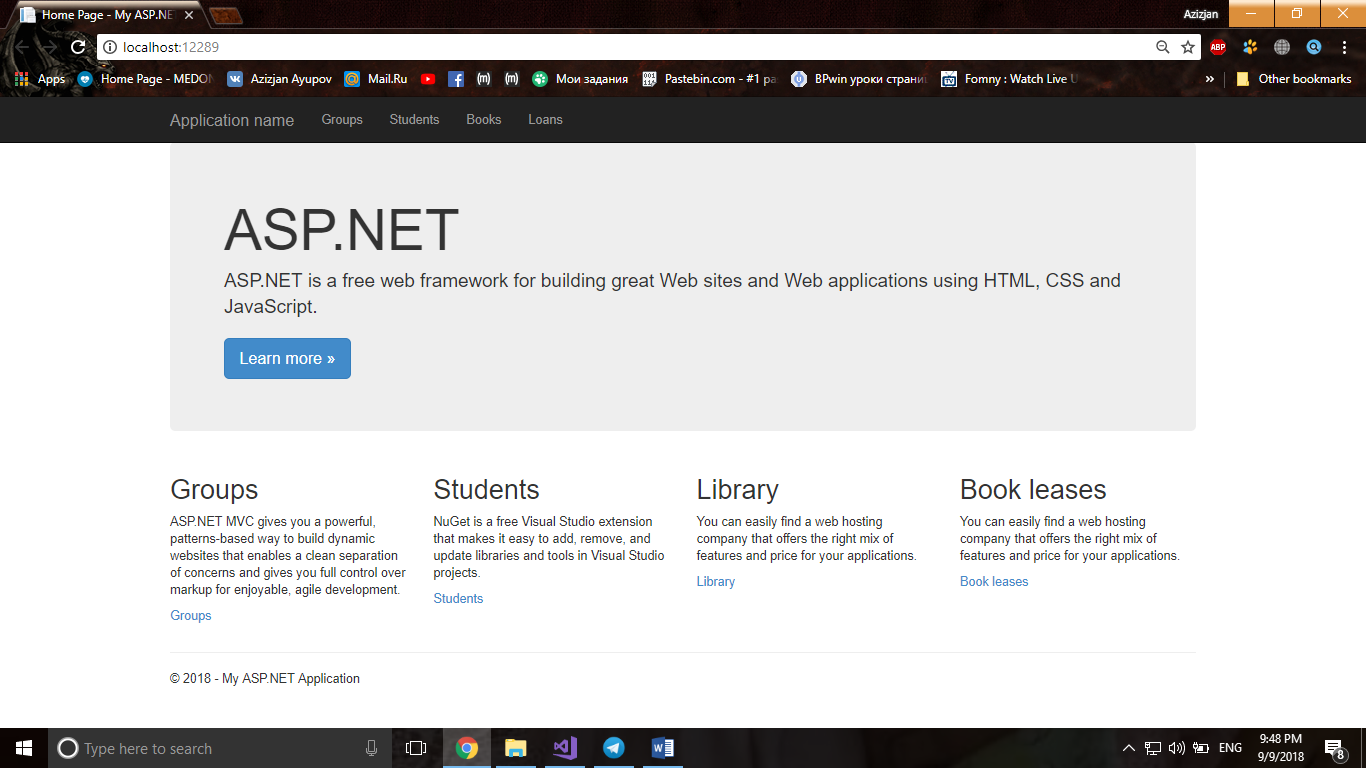
return View(leases);

}

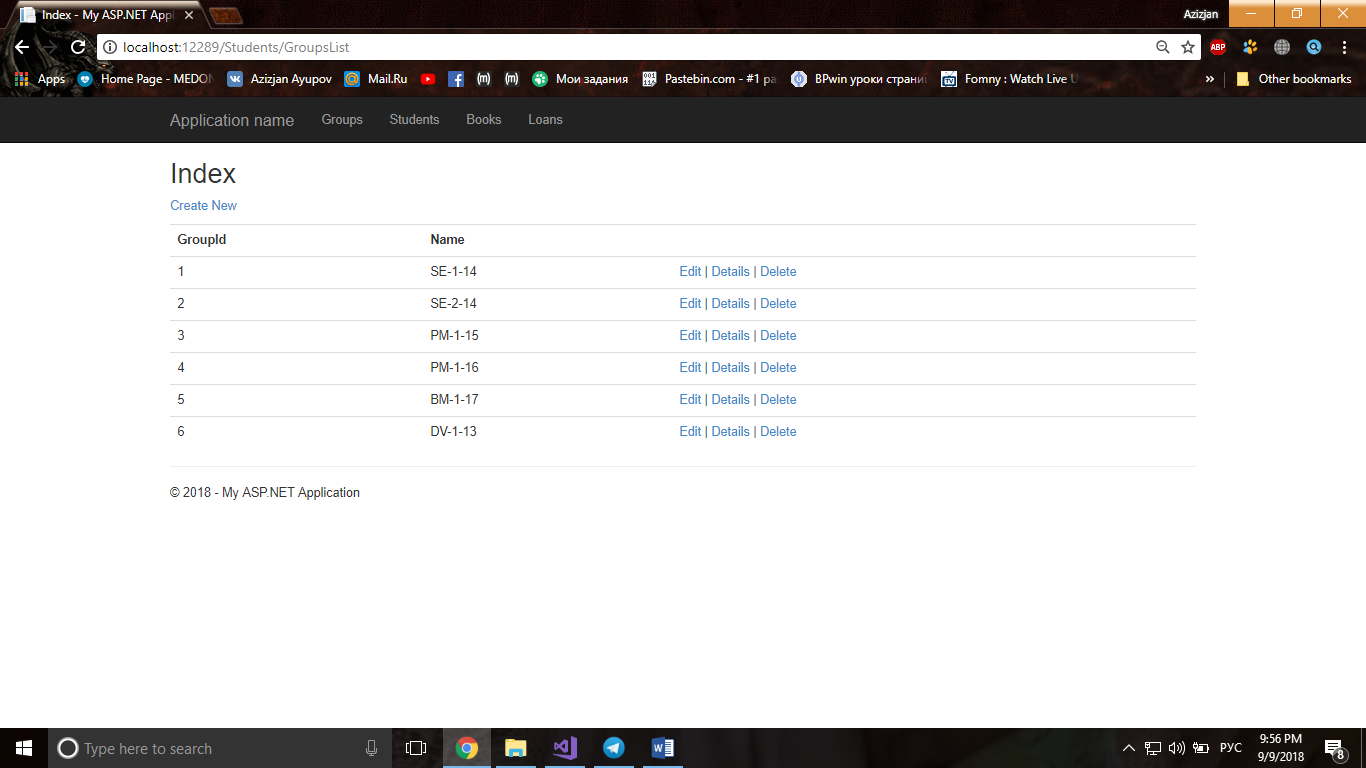
}

## Скриншоты

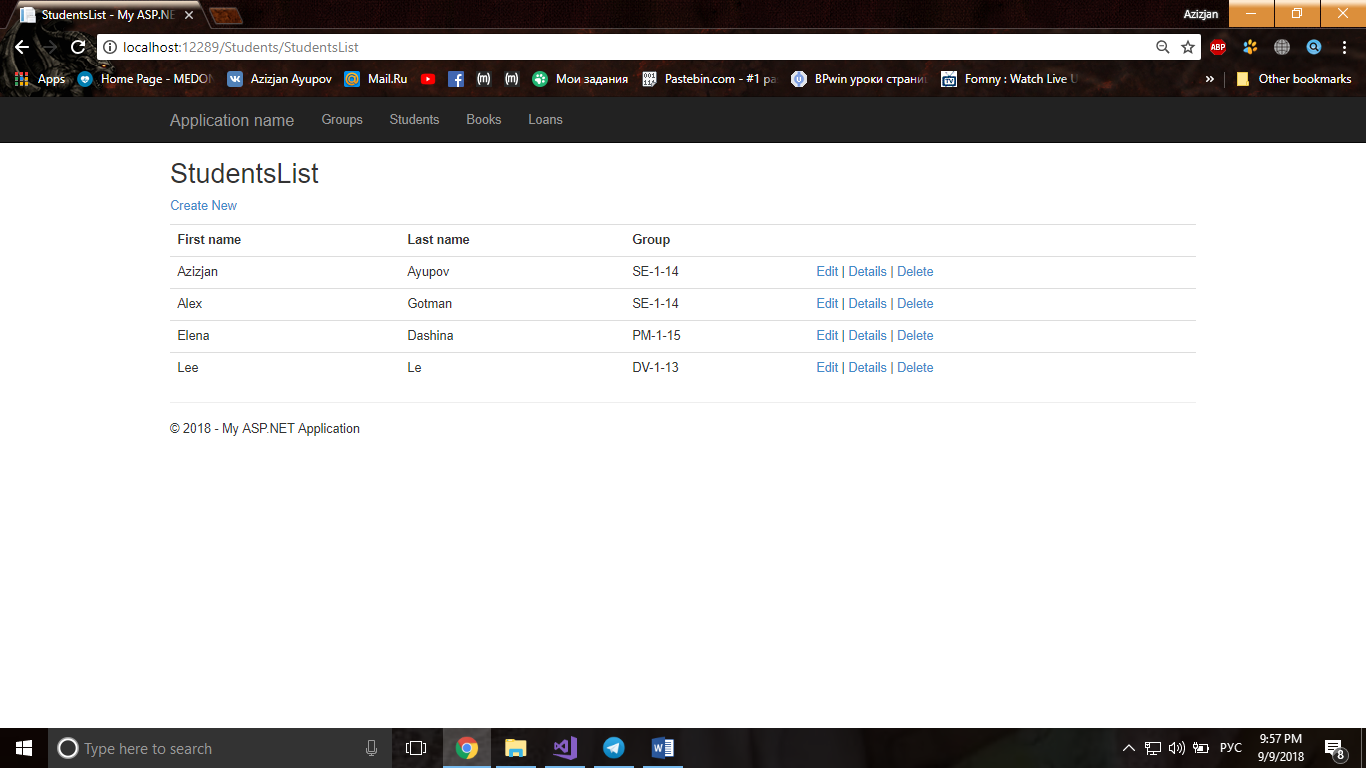
#### Главная



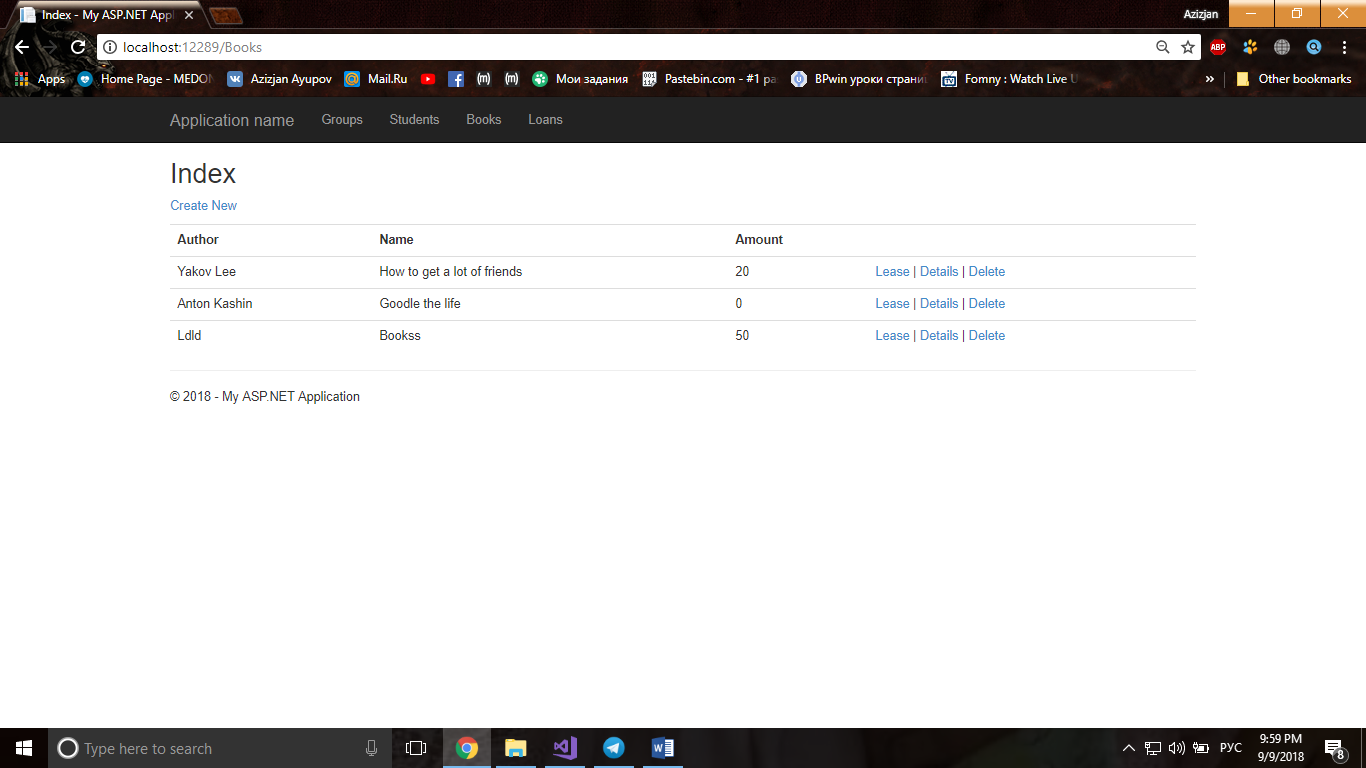
#### Управление группами



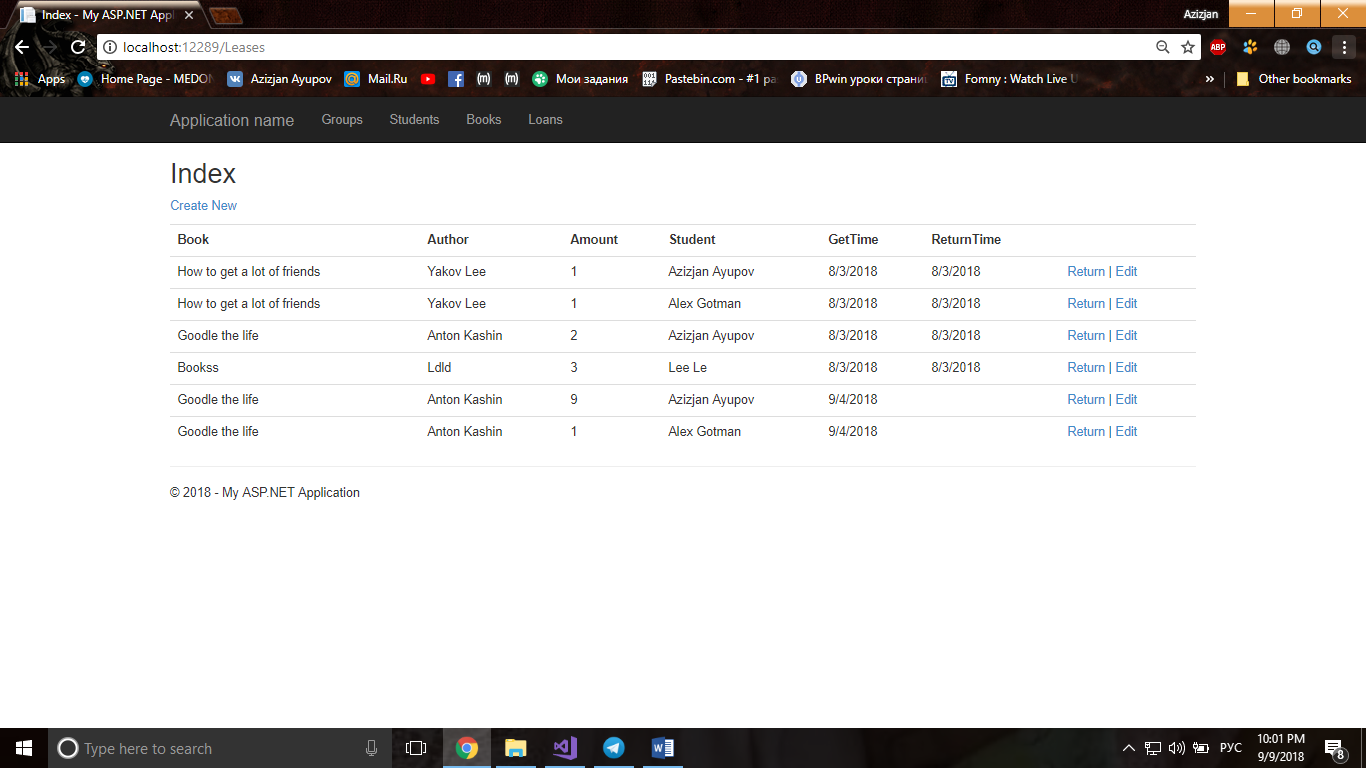
#### Управление студентами



#### Управление книгами



#### Управление заёмами



# Тестирование

Создадим группу ПМ-16,

|  |  |
| --- | --- |
|  |  |

в которую зачислим студента Седокину Елену.

|  |  |
| --- | --- |
|  |  |

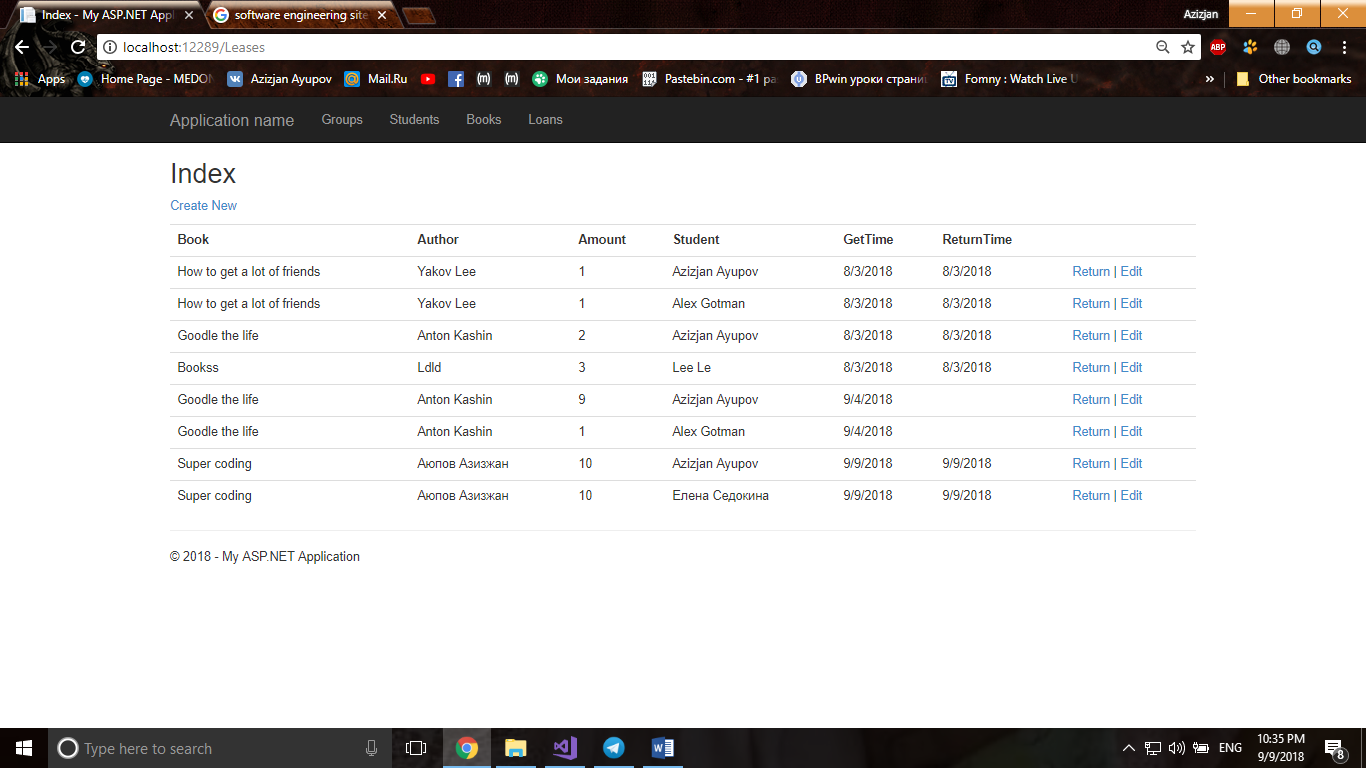
После чего создадим книгу Super Coding за авторством Аюпова Азизжана, якобы пришедшую в количестве 10ти штук.

|  |  |
| --- | --- |
|  |  |

Далее выдадим нашей студентке все десять штук (предположим она берет на всю группу).

|  |  |
| --- | --- |
|  |  |

В завершении примем книги назад.



# Заключение

Была проведена работа по созданию многоуровневой системы, разделяющую уровень доступа к данным, уровень бизнес-логики и уровень представления по разным проектам. Подобная архитектура позволяет как реализовывать большие проекты, так и при разработке определённого уровня, абстрагироваться от специфики реализации остальных уровней за счёт передачи данных через DTO-модели.

# Источники

1. Учебный курс «Web-программирование (С#Java)»
2. Курс «Многоуровневая архитектура» на сайте METANIT.COM https://metanit.com/sharp/mvc5/23.5.php