Практическая работа № 4.  
ASP.Net Чистая архитектура

**Цель практическая работы**

Закрепление теоретических знаний по основам разработки приложений на C# с помощью технологии ASP.NET в соответствии с подходом «Чистая архитектура».

**Постановка задачи**

ASP.NET — это платформа для разработки веб-приложений, созданная компанией Microsoft. Она позволяет разработчикам создавать динамические веб-сайты, веб-приложения и веб-сервисы. ASP.NET поддерживает языки .NET, такие как C#, и предоставляет мощные инструменты и библиотеки для упрощения разработки.

**Основные компоненты ASP.NET**

1. **ASP.NET Web Forms**: Позволяет создавать веб-приложения с использованием событийно-ориентированной модели, аналогичной Windows-приложениям. Подходит для быстрого создания приложений с минимальным количеством кода.
2. **ASP.NET MVC**: Архитектурный паттерн Model-View-Controller, который разделяет приложение на три основных компонента:
   * **Model**: Представляет данные и бизнес-логику.
   * **View**: Отвечает за отображение данных пользователю.
   * **Controller**: Обрабатывает пользовательские запросы, взаимодействует с моделью и выбирает представление для отображения.
3. **ASP.NET Web API**: Позволяет создавать RESTful сервисы, которые могут быть использованы для взаимодействия с клиентскими приложениями, такими как мобильные приложения и одностраничные приложения (SPA).
4. **ASP.NET Core**: Современная версия ASP.NET, которая является кросс-платформенной, высокопроизводительной и более легковесной. Поддерживает создание как веб-приложений, так и API.

В рамках чистой архитектуры центральным элементом приложения являются его бизнес-логика и модель.

В этом случае бизнес-логика не зависит от доступа к данным или другим инфраструктурам, то есть стандартная зависимость инвертируется: инфраструктура и детали реализации зависят от ядра приложения.

Эта функциональность достигается путем определения абстракций или интерфейсов в ядре приложения, которые реализуются типами, определенными в слое инфраструктуры. Такую архитектуру обычно рисуют в виде серии окружностей с общим центром, которая внешне напоминает срез луковицы (рисунок 1).

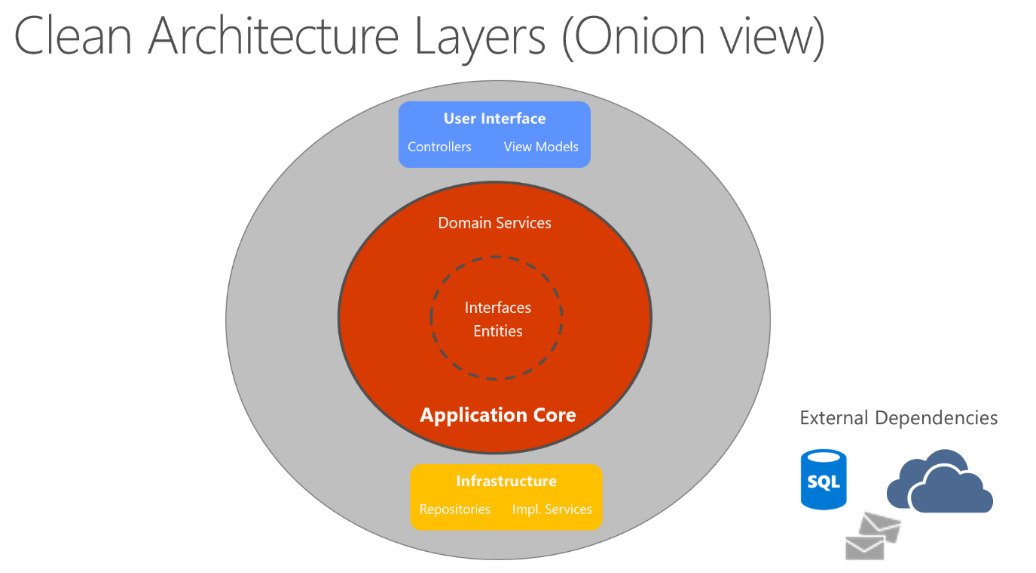
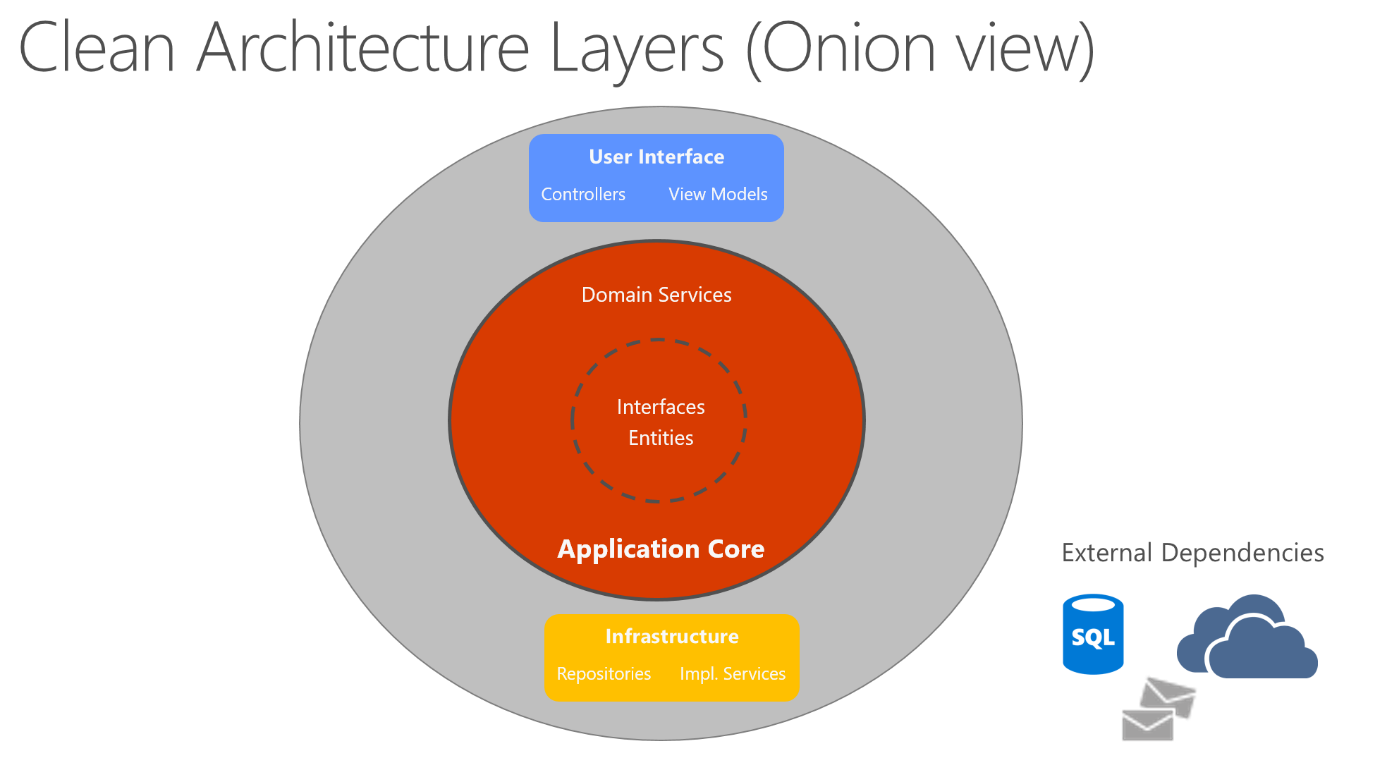
 

Рисунок 1 – Общая схема N-слойной архитектуры

Чистую архитектуру можно описать фразой **- строй приложение вокруг бизнес-логики.** Схема архитектуры представлена на рисунке 2.



Рисунок 2 – Схема «Чистая архитектура»

**Entities**

* Сущности, которые содержат самую важную логику/алгоритмы
* **Пример:** расчет угла броска снаряда

**Use Cases**

* + Наши бизнес-процессы
  + Этот слой содержит прикладную логику - логику нашей предметной области
  + Занимается тем, что манипулирует Entities
  + **Пример:** приложения для расчета угла удара в бильярде и траектории выстрела из пушки

**Контроллеры/Презентеры**

* + Те, с кем внешний мир взаимодействует
  + **Пример:** MVC контроллеры, заглушка для HTTP, модель представления (MVVM)

**UI, БД, Устройства**

* + Это тот самый внешний мир
  + **Пример:** база данных, внешний сервис кеширования или логирования

При правильной реализации поток управления выглядит так:

**Controller -> Use Case Interactor -> Presenter**.

**Пример:**

1. Клиент сделал запрос на контроллер;
2. Контроллер вызвал нужный метод на доменном сервисе;
3. Доменный сервис вызвал метод хранилища, который обновил базу данных.

**Задание на практическую работу**

1. Провести анализ предметной области:

* Определить сущности предметной области согласно варианту.
* Описать свойства каждой сущности, типы данных и ограничения.
* Определить связи между сущностями (1 к 1, один-ко-многим, многие-ко-многим).

1. Спроектировать слои приложения по Clean Architecture:

* Core (доменная логика):
  + - Entities
    - Interfaces
* Infrastructure (реализация репозиториев, работа с БД):
  + - Repositories
    - DbContext
* Web **API** (контроллеры, DTOs):
  + - Controllers
    - Models (DTOs)

1. Определить зависимости между слоями (схематично).
2. Спроектировать API:

* Определить эндпоинты для CRUD-операций (для каждой сущности).
* Описать форматы запросов и ответов (JSON-примеры).
* Указать коды статусов HTTP для разных сценариев.

1. Спроектировать схему работы с БД:

* Выбрать ORM (Entity Framework Core, Dapper).
* Описать структуру DbContext.

1. Разработать план тестирования

* Определить тестовые сценарии для каждого эндпоинта.
* Указать ожидаемые результаты.

**Варианты заданий на лабораторную работу**

Можно придумать свой **уникальный** вариант.

**1. Система учета книг в библиотеке**

* **Класс Book** (книга) – свойства: Id, Title, Author, ISBN, YearPublished, IsAvailable
* **Класс Reader** (читатель) – свойства: Id, Name, Email, Phone, BorrowedBooks (список)
* **Класс Loan** (выдача книги) – свойства: Id, BookId, ReaderId, LoanDate, ReturnDate

**2. Система управления задачами (To-Do List)**

* **Класс Task** (задача) – свойства: Id, Title, Description, DueDate, Priority, IsCompleted
* **Класс Project** (проект) – свойства: Id, Name, Description, Tasks (список задач)
* **Класс User** (пользователь) – свойства: Id, Username, Email, AssignedTasks (список)

**3. Система онлайн-заказов в кафе**

* **Класс MenuItem** (позиция меню) – свойства: Id, Name, Description, Price, Category
* **Класс Order** (заказ) – свойства: Id, CustomerName, OrderDate, Items (список), TotalPrice
* **Класс Employee** (сотрудник) – свойства: Id, Name, Position, AssignedOrders (список)

**4. Система учета студентов и курсов**

* **Класс Student** (студент) – свойства: Id, Name, Email, EnrollmentDate, Courses (список)
* **Класс Course** (курс) – свойства: Id, Title, Description, Instructor, Credits
* **Класс Enrollment** (запись на курс) – свойства: Id, StudentId, CourseId, Grade

**5. Система управления складом**

* **Класс Product** (товар) – свойства: Id, Name, Category, QuantityInStock, Price
* **Класс Supplier** (поставщик) – свойства: Id, Name, ContactInfo, SuppliedProducts (список)
* **Класс InventoryTransaction** (движение товара) – свойства: Id, ProductId, Quantity, TransactionType, Date

**6. Система бронирования билетов в кино**

* **Класс Movie** (фильм) – свойства: Id, Title, Genre, Duration, Showtimes (список сеансов)
* **Класс Ticket** (билет) – свойства: Id, MovieId, SeatNumber, Price, BookingTime
* **Класс Customer** (клиент) – свойства: Id, Name, Phone, BookedTickets (список)

**7. Система учета посещений фитнес-клуба**

* **Класс Member** (клиент) – свойства: Id, Name, MembershipType, JoinDate, Visits (список)
* **Класс WorkoutSession** (тренировка) – свойства: Id, TrainerId, Time, Duration, MaxParticipants
* **Класс Visit** (посещение) – свойства: Id, MemberId, SessionId, CheckInTime

**8. Система поддержки IT-заявок**

* **Класс Ticket** (заявка) – свойства: Id, Title, Description, Status, Priority, CreatedBy
* **Класс Employee** (сотрудник IT) – свойства: Id, Name, Specialization, AssignedTickets (список)
* **Класс Department** (отдел) – свойства: Id, Name, Location, SubmittedTickets (список)

**9. Система аренды автомобилей**

* **Car** (автомобиль) – Id, Model, Year, LicensePlate, RentalPricePerDay, IsAvailable
* **Customer** (клиент) – Id, Name, DriverLicenseNumber, Phone, RentalHistory
* **Rental** (аренда) – Id, CarId, CustomerId, StartDate, EndDate, TotalCost

**10. Система электронного голосования**

* **Election** (выборы) – Id, Title, StartDate, EndDate, Candidates (список)
* **Candidate** (кандидат) – Id, Name, Party, VotesCount
* **Vote** (голос) – Id, VoterId, CandidateId, Timestamp

**11. Система мониторинга пациентов в больнице**

* **Patient** (пациент) – Id, Name, MedicalHistory, CurrentCondition, AssignedDoctorId
* **Doctor** (врач) – Id, Name, Specialization, Patients (список)
* **MedicalRecord** (запись) – Id, PatientId, Diagnosis, Prescription, Date

**12. Система учета ремонта техники**

* **Device** (устройство) – Id, Type, Brand, FaultDescription, OwnerId
* **RepairOrder** (заказ) – Id, DeviceId, TechnicianId, Status, Cost, CompletionDate
* **Technician** (мастер) – Id, Name, Specialization, AssignedOrders

**13. Система управления мероприятиями**

* **Event** (мероприятие) – Id, Name, Date, Location, MaxAttendees, OrganizerId
* **Participant** (участник) – Id, Name, Email, RegisteredEvents (список)
* **Ticket** (билет) – Id, EventId, ParticipantId, PurchaseDate, Price

**14. Система учета домашних животных**

* **Pet** (питомец) – Id, Name, Species, Breed, OwnerId, VaccinationRecords
* **Owner** (владелец) – Id, Name, Address, Phone, Pets (список)
* **VetVisit** (визит к ветеринару) – Id, PetId, VetId, Date, Diagnosis

**15. Система бронирования отелей**

* **HotelRoom** (номер) – Id, RoomNumber, Type, PricePerNight, IsBooked
* **Guest** (гость) – Id, Name, PassportNumber, Phone, BookingHistory
* **Reservation** (бронирование) – Id, RoomId, GuestId, CheckInDate, CheckOutDate

**16. Система учета расходов личных финансов**

* **Transaction** (транзакция) – Id, Amount, Category, Date, Description, UserId
* **User** (пользователь) – Id, Name, Email, MonthlyBudget, Transactions
* **BudgetCategory** (категория) – Id, Name, MonthlyLimit, UserId

**17. Система управления учебными группами**

* **StudentGroup** (группа) – Id, Name, Course, Students (список)
* **Teacher** (преподаватель) – Id, Name, Subject, AssignedGroups
* **Schedule** (расписание) – Id, GroupId, DayOfWeek, Time, Classroom

**18. Система учета доставки еды**

* **Restaurant** (ресторан) – Id, Name, CuisineType, MenuItems (список)
* **Order** (заказ) – Id, CustomerId, RestaurantId, DeliveryAddress, TotalPrice
* **DeliveryDriver** (курьер) – Id, Name, VehicleType, AssignedOrders

**19. Система управления проектами в IT**

* **Project** (проект) – Id, Title, Description, Deadline, Status, Team
* **TeamMember** (участник) – Id, Name, Role, AssignedTasks
* **Task** (задача) – Id, Title, Priority, Status, AssigneeId, ProjectId

**20. Система учета аренды жилья**

* **Apartment** (квартира) – Id, Address, RentPrice, Bedrooms, IsOccupied
* **Tenant** (арендатор) – Id, Name, ContactInfo, LeaseHistory
* **Lease** (договор) – Id, ApartmentId, TenantId, StartDate, EndDate

**21. Система управления контентом блога**

* **BlogPost** (статья) – Id, Title, Content, AuthorId, PublishDate, Tags
* **Author** (автор) – Id, Name, Bio, Posts (список)
* **Comment** (комментарий) – Id, PostId, AuthorName, Text, Timestamp

**22. Система учета спортивных соревнований**

* **Athlete** (спортсмен) – Id, Name, Sport, Team, Competitions
* **Competition** (соревнование) – Id, Name, Date, Location, Results
* **Result** (результат) – Id, AthleteId, CompetitionId, Score, Rank

**23. Система мониторинга транспорта**

* **Vehicle** (транспорт) – Id, Type, LicensePlate, CurrentLocation, Status
* **Driver** (водитель) – Id, Name, LicenseNumber, AssignedVehicleId
* **Route** (маршрут) – Id, VehicleId, StartPoint, EndPoint, Distance

**24. Система учета игр в киберспорте**

* **Game** (игра) – Id, Title, Genre, ReleaseDate, Teams (список)
* **Team** (команда) – Id, Name, Players, TournamentsWon
* **Match** (матч) – Id, Team1Id, Team2Id, WinnerId, Score

**Пример разработки приложения**

**Тема:** Система управления библиотекой (сущности Book, Author, Genre)

**Шаг 1. Анализ предметной области**

**1.1. Определение сущностей (рисунок 3)**

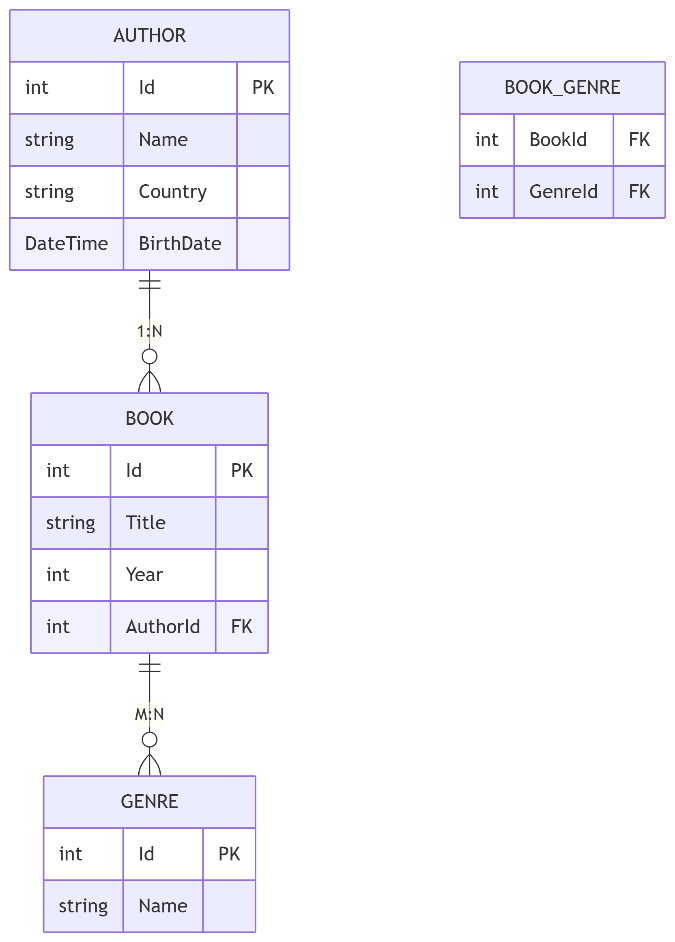
****

Рисунок 3 – Схема предметной области

**1.2. Описание сущностей**

| **Сущность** | **Поля** | **Описание** |
| --- | --- | --- |
| Author | Id, Name, Country | Автор книг |
| Book | Id, Title, Year, Author | Книга с указанием автора |
| Genre | Id, Name | Жанр литературы |

**Шаг 2. Проектирование слоёв**

**2.1. Структура решения**

LibrarySystem/

├── src/

│ ├── LibrarySystem.Core/

│ │ ├── Entities/

│ │ ├── Interfaces/

│ │ └── Services/

│ ├── LibrarySystem.Infrastructure/

│ │ ├── Data/

│ │ ├── Repositories/

│ │ └── Services/

│ └── LibrarySystem.Web/

│ ├── Controllers/

│ ├── DTOs/

│ └── Middleware/

└── tests/

**Шаг 3. Проектирование API**

**3.1. Эндпоинты для Book**

### Получить все книги

GET {{host}}/api/books

### Получить книгу по ID

GET {{host}}/api/books/1

### Создать книгу

POST {{host}}/api/books

Content-Type: application/json

{

"title": "Clean Code",

"year": 2008,

"authorId": 1,

"genreIds": [1, 2]

}

**3.2. Пример ответа**

{

"id": 1,

"title": "Clean Code",

"year": 2008,

"author": {

"id": 1,

"name": "Robert Martin"

},

"genres": [

{"id": 1, "name": "Programming"},

{"id": 2, "name": "Best Practices"}

]

}

**Шаг 4. Работа с БД**

**4.1. DbContext**

public class AppDbContext : DbContext

{

public DbSet<Book> Books { get; set; }

public DbSet<Author> Authors { get; set; }

public DbSet<Genre> Genres { get; set; }

public DbSet<BookGenre> BookGenres { get; set; }

}

**4.2. Миграции**

*# Создание миграции*

dotnet ef migrations add InitialCreate -p LibrarySystem.Infrastructure -s LibrarySystem.Web

*# Применение миграций*

dotnet ef database update

**Шаг 5. План тестирования**

**5.1. Тест-кейсы для Book**

| **Сценарий** | **Метод** | **Ожидаемый результат** |
| --- | --- | --- |
| Создание книги без автора | POST /api/books | 400 Bad Request |
| Получение несуществующей книги | GET /api/books/999 | 404 Not Found |
| Удаление книги с жанрами | DELETE /api/books/1 | 200 OK (каскадное удаление) |

**Шаг 6. Дополнительные настройки**

**6.1. docker-compose.yml**

services:

db:

image: postgres:latest

environment:

POSTGRES\_PASSWORD: mysecretpassword

ports:

- "5432:5432"

volumes:

- pgdata:/var/lib/postgresql/data

volumes:

pgdata:

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»