**Раздел 1.** Практическая значимость разрабатываемого ПС

Данная программа разрабатывается для автоматизации управления салоном красоты. Она позволяет управлять списком услуг и мастеров, записывать клиентов на услуги, формировать финансовые отчеты и отчеты по сотрудникам, вести учет посещений клиентов

Это помогает администраторам салона эффективно управлять записями клиентов и тратить меньше времени на создание отчетности.

Процессы управления салоном красоты до и после автоматизации:

1. Управление услугами и мастерами

До автоматизации: администратор вручную записывает название услуги, ее стоимость и имена мастеров, которые ее оказывают, в бумажный журнал или таблицу Excel. Поиск нужной услуги в журнале может занимать время, особенно если список большой.

На каждую операцию с услугой (добавление, изменение, просмотр) тратится от 2 до 10 минут администратора, в зависимости от сложности записи и размера журнала. В неделю это может составлять до 1-2 часов рабочего времени.

После автоматизации: администратор через интерфейс программы вводит название, стоимость и список мастеров. Система автоматически проверяет корректность ввода. Информацию можно сохранить в txt-файле services. Добавление услуги занимает около 30-60 секунд. Проверка ввода автоматизирована. Предусмотрена возможность отображения всех услуг, это занимает менее 5 секунд

Пример файла services.txt:

стрижка1;500;Петр,Олег,Иван

стрижка2;300;Петр,Влад,Жора

стрижка3;400;Дмитрий,Анна,Алексей

1. Запись клиентов

До автоматизации: администратор открывает журнал записей, ищет доступное время, уточняет у клиента желаемую услугу и мастера. Записывает имя клиента, услугу, мастера, дату и время. Администратор должен убедиться, что клиент уже существует или корректно записать его данные, а также что выбранная услуга существует в прайс-листе. Создание одной записи может занимать от 5 до 15 минут, в зависимости от занятости администратора, количества клиентов в очереди и сложности запроса. Ежедневно это может составлять 1-3 часа рабочего времени.

Присутствует высокая вероятность ошибок, таких как неправильная запись времени, дублирование записей, пропуск клиентов, неверное указание услуги или мастера.

После автоматизации: система предоставляет интерфейс, где администратор вводит имя клиента. Программа, мгновенно находит клиента или создает нового. Список услуг и мастеров отображается автоматически. Система предотвращает выбор несуществующих услуг или мастеров. Создание записи занимает около 1 минуты. Информацию можно сохранить в txt-файле appointments.

Пример файла appointments.txt:

Жора;стрижка1;Петр;2025-05-27

Андрей;стрижка2;Влад;2025-05-30

1. Финансовые отчеты и статистика

До автоматизации: Администратор вручную суммирует стоимость всех оказанных услуг за определенный период, просматривая записи в журналах.  Подсчитывает количество посещений каждого клиента, суммируя записи. Суммирует стоимость услуг, оказанных каждым мастером. Составление даже простого отчета может занимать от 30 минут до нескольких часов, в зависимости от объема данных и сложности анализа.

Риск ошибок очень высок. Ручные вычисления подвержены ошибкам, человеческий фактор. Анализ эффективности работы салона затруднен.

После автоматизации: Пользователь выбирает тип отчета (клиенты, выручка, сотрудники) и указывает период. Система автоматически обрабатывает данные из списков и выводит отчет на экран. Формирование отчета занимает от 10 до 30 секунд. Отчеты сохраняются в файлы (client\_rating.txt, revenue\_report.txt, employee\_report.txt)

Примеры отчетов:

Рейтинг клиентов за период 01.01.2025 - 31.01.2025:

Иванов: 5 посещений

Петрова: 3 посещения

Сидоров: 2 посещения

Выручка за период 01.01.2025 - 31.01.2025: 45,000 руб

Отчет по сотрудникам:

Иванова: 25,000 руб

Петрова: 15,000 руб

Сидорова: 5,000 руб

Без данной программы учет услуг и записей велся бы в бумажных журналах или простых электронных таблицах. Расчет выручки и статистики выполнялся бы вручную, что сопряжено с высоким риском ошибок при бронировании записей и сложностями с анализом эффективности работы салона. Ручной ввод и обработка данных требуют значительных временных ресурсов, а ошибки могут привести к потере клиентов и снижению доходов. Разрабатываемая программа устраняет эти недостатки, предоставляя надежный и эффективный инструмент для управления бизнесом.

**Раздел 2.** Функциональные требования

Основные функции программы:

1. Добавление услуг:

Пользователь может ввести информацию о новой доступной услуге, то есть сначала пользователь указывает название, затем стоимость и список мастеров. В программе предусмотрена проверка вводимой информации. Это позволяет легко добавлять новые услуги при необходимости.

Пример использования:

* + Выбрать "1. Управление услугами"
  + Откроется меню услуг
  + Выбрать "2. Добавить услугу"
  + Ввести данные: "Стрижка", "1500", "Иванова,Петрова"
* Подтвердить сохранение

1. Просмотр всего списка услуг:

Программа предоставляет возможность просмотра списка всех услуг.

Пример использования:

* Выбрать "1. Управление услугами"
* Откроется меню услуг
* Выбрать "1. Показать все услуги"

1. Сохранение в файл services.txt:

Реализована возможность сохранения услуг в txt-файл services. Это помогает сохранять информацию между сеансами. Если при сохранении будет допущена ошибка программа уведомит пользователя об этом.

Пример использования:

* Выбрать "1. Управление услугами"
* Откроется меню услуг
* Выбрать "3. Сохранить в файл"

1. Запись клиентов:

Пользователь может ввести имя клиента, выбрать доступную услугу, доступных мастеров для конкретной услуги и дату. Это позволяет легко добавлять новые записи.

Пример использования:

* Выбрать "2. Запись клиентов"
* Откроется меню записи клиентов
* Выбрать "1. Новая запись"
* Ввести имя клиента: "Иванов"
* Выбрать услугу из списка
* Выбрать мастера из доступных
* Ввести дату: "2023-12-15"

1. Просмотр списка всех записей:

Пользователь может просмотреть список всех записей.

* Выбрать "2. Запись клиентов"
* Откроется меню записи клиентов
* Выбрать "2. Список всех записей"

1. Сохранение записей в файл appointments.txt

Реализована возможность сохранения услуг в txt-файл appointments. Это помогает сохранять информацию между сеансами. Если при сохранении будет допущена ошибка программа уведомит пользователя об этом.

* Выбрать "2. Запись клиентов"
* Откроется меню записи клиентов
* Выбрать "3. Сохранить записи в файл"

1. Составление отчетности:

Пользователь может создать 3 отчета, отчет по клиентам, отчет по доходам и отчет по сотрудникам. Отчет по клиентам возвращает информацию о том сколько посещений было у всех клиентов. Отчет по сотрудникам показывает сколько прибыли принес конкретный сотрудник. Отчет по доходам возвращает отчет по всей прибыли. Предусмотрена проверка на правильность ввода даты.

**Раздел 3. Модульная структура**

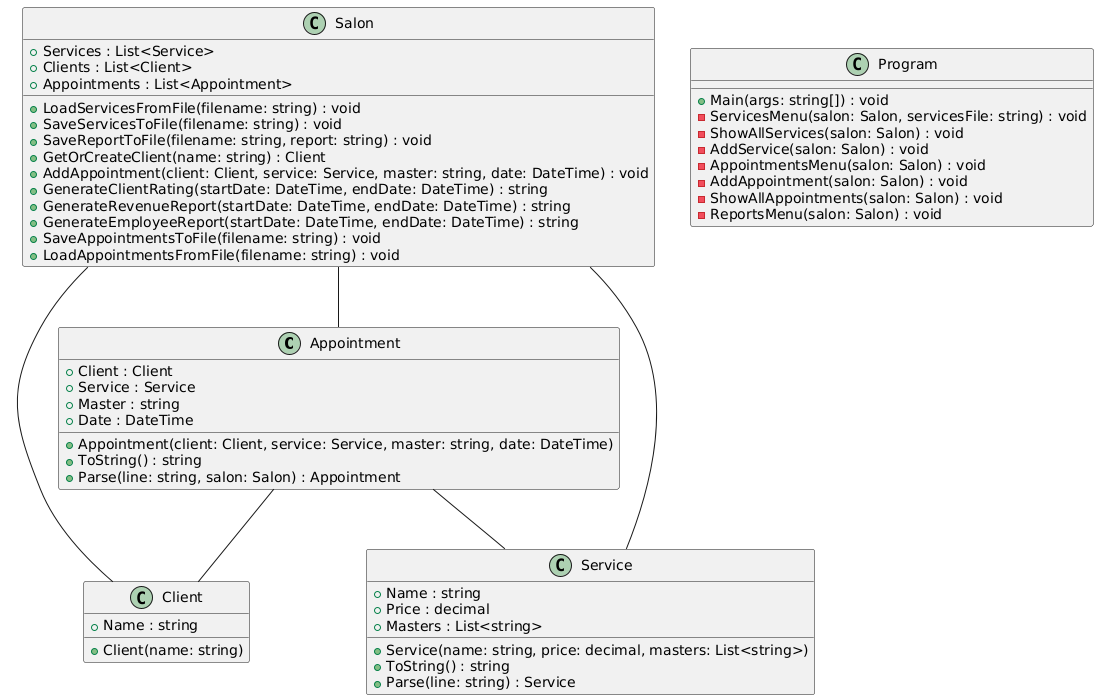


Рисунок 1 – Диаграмма классов

Класс Salon

Класс управляющий всей бизнес-логикой салона. Координирует работу всех остальных классов.

Методы:

LoadServicesFromFile() - Загружает услуги из файла при запуске

SaveServicesToFile() - Сохраняет все услуги в файл

SaveReportToFile() – Сохраняет отчет в файл

GetOrCreateClient() - Находит существующего или создает нового клиента

AddAppointment() - Добавляет новую запись на услугу

GenerateClientRating() - Формирует рейтинг клиентов по посещениям

GenerateRevenueReport() - Создает отчет о выручке за период

GenerateEmployeeReport() - Генерирует отчет по доходам мастеров

LoadAppointmentsFromFile() - Загружает записи из файла при запуске

SaveAppointmentsToFile() - Сохраняет все записи в файл

Класс Service

Модель услуги. Описывает услугу салона с ее характеристиками. Основа для бронирования и расчетов.

Поля:

Name(string) - Название услуги (стрижка, маникюр)

Price(decimal) - Стоимость услуги

Masters(list) - Список мастеров, оказывающих услугу

Методы:

ToString() – Переопределенный метод, форматирует данные для сохранения в файл

Parse() - Преобразует строку из файла в объект Service

Класс Client

Модель клиента. Хранит информацию о клиентах салона. Используется для учета посещений.

Поля:

Name(string) - Имя клиента (уникальный идентификатор)

Класс Appointment

Модель записи. Связывает клиента, услугу и мастера на конкретную дату. Основа для учета и отчетности.

Поля:

Client (string) - Клиент записи

Service (Service) - Выбранная услуга

Master (string) - Мастер, выполняющий услугу

Date (DateTime) - Дата и время записи

Методы:

ToString() – Переопределенный метод, форматирует данные для сохранения в файл

Parse() - Преобразует строку из файла в объект Appointment

Класс Program

Пользовательский интерфейс. Обеспечивает взаимодействие с пользователем через консоль. Координирует работу меню.

Методы:

Main() - Точка входа в программу

ServicesMenu() - Меню управления услугами

AppointmentsMenu() - Меню записи клиентов

ReportsMenu() - Меню отчетности

ShowAllServices() - Отображает все услуги

AddService() - Добавляет новую услугу

AddAppointment() - Создает новую запись

ShowAllAppointments() - Показывает все записи

**Критерии выделения класссов**

1. Класс Appointment (Запись)

Необходим для представления отдельной записи в салоне красоты. Это ключевой “бизнес-объект”, который связывает клиента, услугу, мастера и дату/время.

Отвечает за хранение информации об отдельной записи, инкапсуляцию связанных данных (клиент, услуга, мастер, дата).

Нам нужен способ представлять и работать с отдельными записями. Без этого класса все данные о записи пришлось бы хранить в отдельных переменных или структурах данных, что сделало бы код сложнее для чтения, поддержки и расширения.

2. Класс Client (Клиент)

Нужен для представления клиента салона.

Отвечает за хранение имени клиента, инкапсуляцию данных клиента.

Без него сложно обойтись, так как нам нужен способ связывать записи с клиентами.

3. Класс Service (Услуга)

Нужен для представления услуги, предлагаемой салоном (например, стрижка, маникюр).

Отвечает за хранение информации об услуге (название, цена и список доступных мастеров), инкапсуляцию данных об услуге.

Без него сложно обойтись, так как нам нужен способ связывать записи с конкретными услугами. Он позволяет хранить и управлять предлагаемыми услугами.

4. Класс Salon (Салон)

Необходим для управления общими данными и операциями салона. Действует как центральная точка для логики салона.

Отвечает за хранение списков услуг, клиентов и записей, загрузку и сохранение данных из файлов (услуги, потенциально клиенты и записи), создание новых клиентов (или получение существующих), добавление записей и генерацию отчетов.

Предоставление интерфейса для других частей приложения для взаимодействия с данными и бизнес-логикой.

Без него сложно обойтись, так как он предоставляет центральную точку для организации данных и инкапсулирует бизнес-логику салона, также обрабатывает файловые операции для сохранения данных.

**Связи между классами**

1. Связь между Salon и Service:

Salon содержит список Service.

Salon управляет списком услуг (Services).

Салон должен знать, какие услуги он предлагает. Эта связь позволяет салону получать доступ ко всем услугам, добавлять новые услуги, редактировать существующие и т.д.

2. Связь между Salon и Client:

Salon содержит список Client.

Salonуправляет списком клиентов (Clients). Как и с услугами, салон должен знать своих клиентов.

Салон должен знать, кто его клиенты. Связь позволяет создавать новых клиентов, находить существующих и хранить информацию о них.

3. Связь между Salon и Appointment:

Salon содержит список Appointment.

Salonуправляет списком записей (Appointments). Салон управляет графиком записей клиентов на услуги.

Салон должен управлять графиком работы, а также хранить информацию о каждой записи.

4. Связь между Appointment и Client:

Appointment использует Client.

Appointment имеет ссылкуна объект Client

Каждая запись должна быть связана с клиентом, который записался на услугу.

5. Связь между Appointment и Service:

Appointment использует Service.

Appointment имеет ссылку на объект Service

Каждая запись должна быть связана с услугой, на которую записался клиент.

**Аргументация выбора типов полей**

Класс Appointment

* Client Client { get; set; }:

Тип: Client (ссылка на объект класса Client).

Ссылка на объект Client, представляющий клиента, который записался на эту встречу.

Связывает встречу с конкретным клиентом. Без этого невозможно узнать, кто записался на конкретную встречу.

Необходим для хранения информации о клиенте, посещающем салон.

* Service Service { get; set; }:

Тип: Service (ссылка на объект класса Service).

Ссылка на объект Service, представляющий услугу, на которую записался клиент.

Связывает встречу с конкретной услугой. Без этого невозможно узнать, на какую услугу была сделана запись.

Необходим для хранения информации об услуге, оказанной клиенту.

* string Master { get; set; }:

Тип: string (строка).

Имя мастера, который будет оказывать услугу.

Позволяет указать, какой мастер будет обслуживать клиента в этой встрече.

Необходим для хранения информации о мастере, оказывающем услугу.

* DateTime Date { get; set; }:

Тип: DateTime (дата и время).

Дата и время, когда запланирована встреча.

Позволяет определить, когда должна состояться встреча.

Необходимо для хранения даты и времени встречи.

**Порядок использования методов**

Создание новой записи на услугу:

1. Program.Main()

Создает объект salon типа Salon. Это точка входа для взаимодействия со всем приложением.

Загружает данные об услугах, используя salon.LoadServicesFromFile().

1. AppointmentsMenu()

Предлагает выбор действий для пользователя

1. AddAppointment

Пользователь вводит имя клиента, выбирает необходимую услуги их доступных, выбирает доступного мастера, вводит дату встречи и запись добавляется в список Appointments.

**Раздел 4. Описание алгоритмов**

Опишем алгоритм добавления услуги

1. Вывести сообщение пользователю, предлагающее ввести название услуги.
2. Прочитать введенное пользователем название услуги.
3. Проверить, что название не пустое (или не состоит только из пробелов). Если пустое, вывести сообщение об ошибке.
4. Вывести сообщение пользователю, предлагающее ввести цену услуги.
5. Прочитать введенную пользователем цену.

6. Попытаться преобразовать введенное значение в число типа decimal.

Если преобразование прошло успешно, перейти к следующему шагу.

Если преобразование не удалось (введено не число), вывести сообщение об ошибке (например, “Неверный формат цены. Введите число.”)

1. Вывести сообщение пользователю, предлагающее ввести список мастеров, разделенных запятыми.
2. Прочитать введенный пользователем список мастеров.
3. Создать новый объект Service, используя полученные данные (название, цена, список мастеров).
4. Добавить созданный объект Service в список salon.Services.
5. Сообщить пользователю, что услуга успешно добавлена.

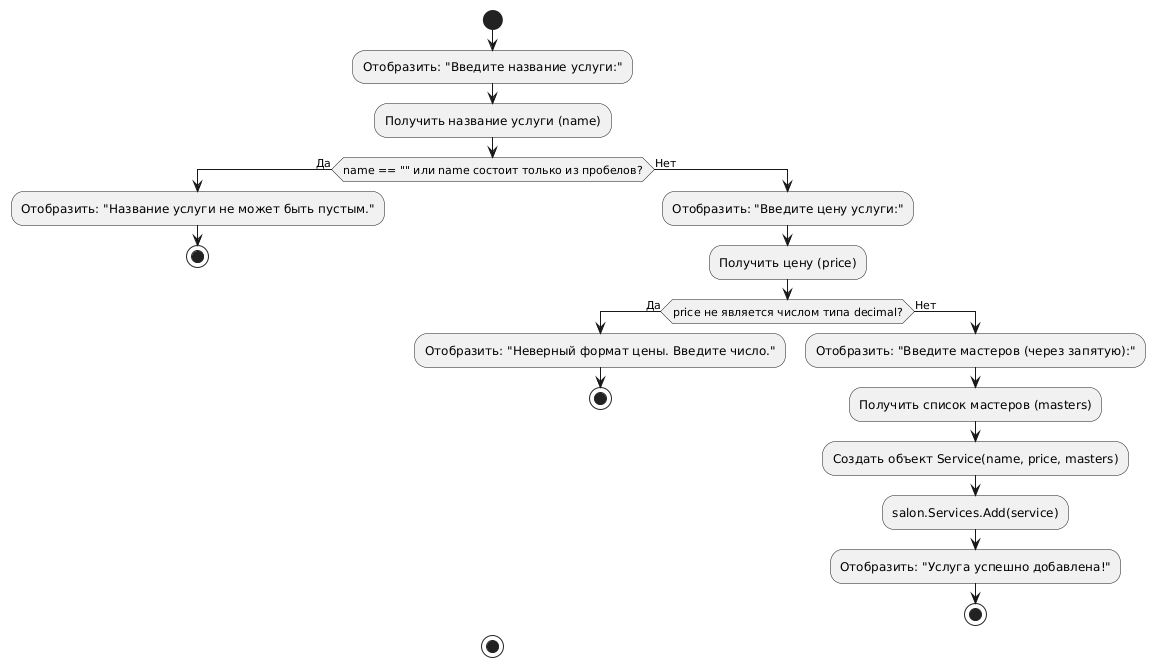


Рисунок 2 – Алгоритм добавления услуги

Давайте рассмотрим еще один немаловажный метод. Добавление записи.

1. Пользователь вводит имя клиента
2. Проверка пустое ли имя, если да, то выдаем ошибку
3. Поиск или создание клиента
4. Программа отображает список существующих услуг, а пользователь выбирает необходимую, если введен неверный формат числа или номер вне диапазона списка услуг, то выдаем ошибку.
5. Программа получает услугу по индексу и отображает список доступных мастеров
6. Пользователь вводит нужного мастера, если данного мастера нет в списке, то появляется ошибка
7. Пользователь вводит дату записи, если указан не правильный формат даты, то появляется ошибка
8. Программа выводит сообщение об успешном добавлении записи

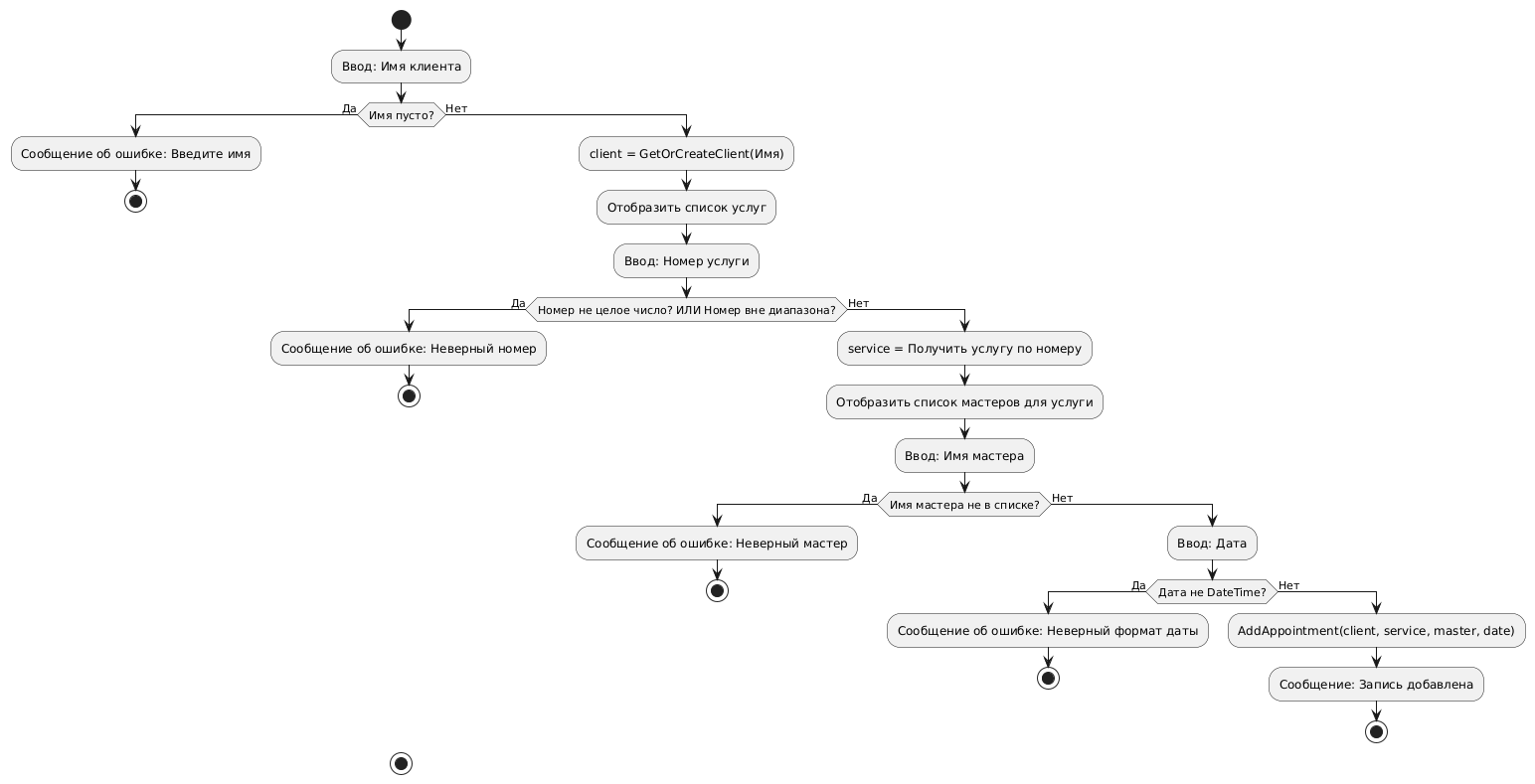


Рисунок 3 – Алгоритм добавления записи

**Раздел 5. Разработка проекта Visual Studio C#**

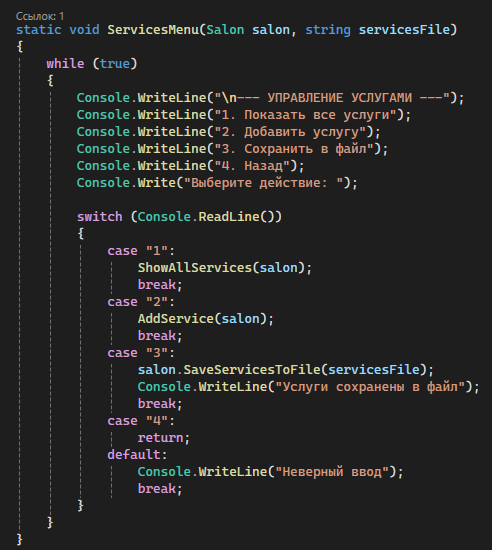


Рисунок 4 – Меню услуг

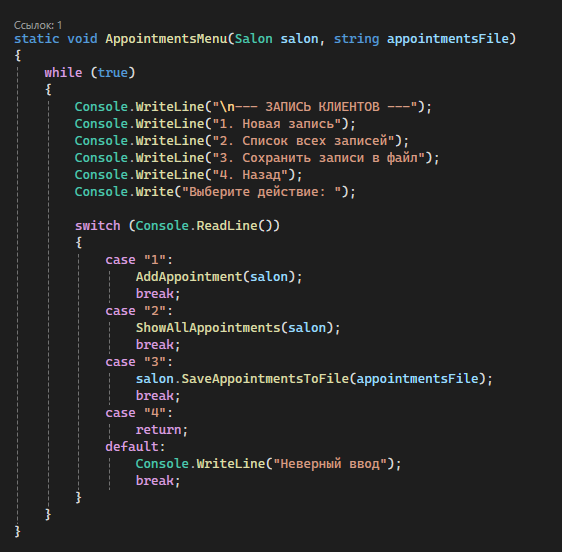


Рисунок 5 – Меню записей



Рисунок 6 – Меню отчетов



Рисунок 7 – Класс Service с переопределенным методом ToString и методом Parse

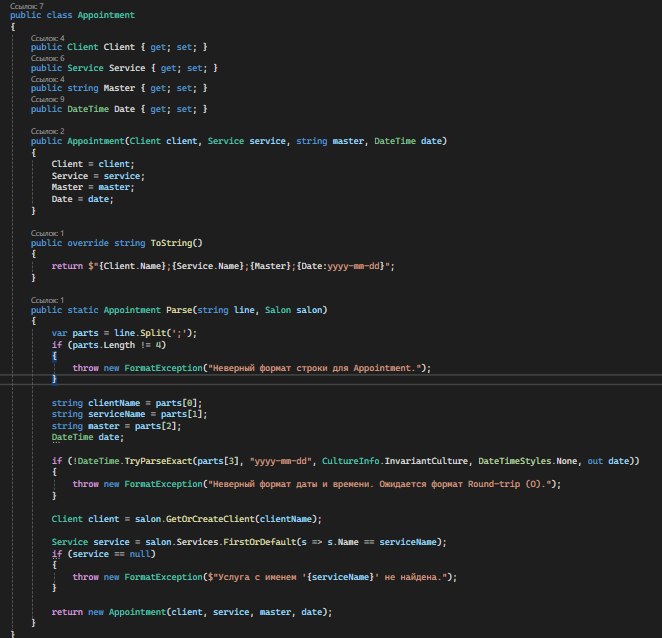


Рисунок 8 – Класс Appointment с переопределенным методом ToString и методом Parse

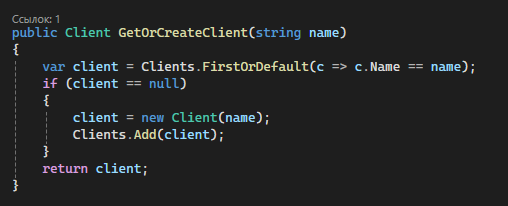


Рисунок 9 – Метод для поиска или создания пользователя

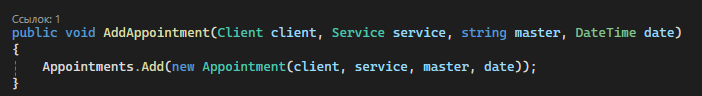


Рисунок 10 – Метод для добавления записи в список

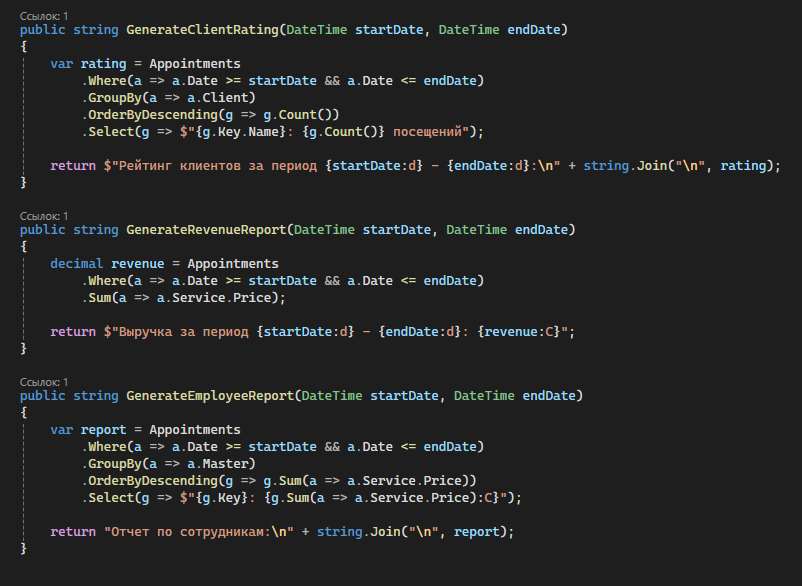


Рисунок 11 – Методы для формирования отчетов