МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Проектирование информационных систем**

**Тема:** «Объектно-ориентированное моделирование. СТРУКТУРНЫЕ ДИАГРАММЫ UML»

**Цель:** Изучение методологии объектно-ориентированного моделирования средствами UML. Ознакомление с основными принципами объектно-ориентированного проектирования программного обеспечения, получение навыков проектирования *структуры* информационной системы с применением UML.

Выполнил:

Кальчевский Даниил Андреевич,

3 курс 1 группа

Минск 2023

**1. Постановка задачи**

В ходе данной лабораторной работы необходимо разработать диаграмму классов и пакетов:

диаграмма классов – визуальное представление структуры системы, описывая классы, их атрибуты, методы и взаимосвязи между классами;

диаграмма пакетов – используется для организации элементов модели в пакеты или группы.

В результате необходимо иметь визуализированные диаграммы, а также отчёт по проделанной лабораторной работе.

**2. Описание программных средств**

Draw.io – это бесплатное онлайн-приложение для создания диаграмм и схем. Оно позволяет пользователям создавать профессионально выглядящие диаграммы благодаря широкому набору инструментов и функций.

Название: Draw.io

Версия: Веб-приложение

Разработчик: JGraph Ltd.

Адрес загрузки: https://www.draw.io/

Режим использования: Онлайн

Доступность на платформах: Веб-браузеры (Google Chrome, Mozilla Firefox, Safari и другие)

Draw.io позволяет создавать различные типы диаграмм, такие как блок-схемы, организационные диаграммы, UML-диаграммы, сетевые диаграммы и многое другое. Он также поддерживает импорт и экспорт файлов в различных форматах, включая PNG, JPEG, PDF и SVG.

Draw.io является мощным инструментом для проектирования и моделирования, который может быть использован в различных областях, включая разработку программного обеспечения, системный анализ, проектирование баз данных и многое другое. Он предоставляет удобный интерфейс и интуитивно понятные инструменты, что делает его доступным для широкого круга пользователей.

Draw.io доступен бесплатно и не требует установки дополнительного программного обеспечения. Он также интегрируется с различными облачными хранилищами, такими как Google Drive, OneDrive и Dropbox, что обеспечивает удобное сохранение и совместную работу над проектами.

**3. Описание практического задания**

В разрабатываемой UML-диаграмме классов присутствуют следующие сущности:  
 Класс DbContext:

* DBCONTEXT – поле контекста базы данных;
* CLIENT – набор данных из таблицы CLIENT;
* ABONEMENT – набор данных из таблицы ABONEMENT ;
* ABONEMENTREQUEST – набор данных из таблицы ABONEMENTREQUEST;
* SALEOFABONEMENT – набор данных из таблицы SALEOFABONEMENT;
* LOGPAS – набор данных из таблицы LOGPAS;
* EMPLOYEE – набор данных из таблицы EMPLOYEE;
* GetContext – метод, возвращающий строку контекста БД;
* OnModelCreating – метод, возвращающий строку модели создания;

Класс CLIENT:

* IDCLIENT – поле уникального идентификатора клиента;
* IDLOGPAS – поле уникального идентификатора авторизации;
* FIRSTNAME – поле фамилии клиента;
* LASTNAME – поле имени клиента;
* MIDDLENAME – поле отчество клиента;
* AGE – поле возраста клиента;
* GENDER – поле пола клиента;
* PHONE – поле телефона клиента;

Класс EMPLOYEE:

* IDEMPLOYEE – поле уникального идентификатора сотрудника;
* IDLOGPAS – поле уникального идентификатора авторизации;
* FIRSTNAME – поле фамилии сотрудника;
* LASTNAME – поле имени сотрудника;
* MIDDLENAME – поле отчество сотрудника;
* PHONE – поле телефона сотрудника;
* SALARY – поле зарплаты сотрудника;

Класс LOGPAS:

* IDLOGPAS – поле уникального идентификатора;
* LOGINVALUE – поле логина;
* PASSWORDVALUE – поле пароля;
* USERTYPE – поле тип пользователя;

Класс ABONEMENT:

* IDABONEMENT – поле уникального идентификатора абонемента;
* PRICEABONEMENT – поле цены абонемента;
* AMOUNTOFVISITS – поле количество посещений;
* IMGABONEMENT – поле картинки абонемента;

Класс SALEOFABONEMENT:

* IDSALE – поле уникального идентификатора продажи;
* IDCLIENT – поле уникального идентификатора клиента;
* DATABEGIN – поле даты начало абонемента;
* DATAEND – поле даты конца абонемента;

Класс SALEOFABONEMENT:

* IDSALE – поле уникального идентификатора продажи;
* IDCLIENT – поле уникального идентификатора клиента;
* DATABEGIN – поле даты начало абонемента;
* DATAEND – поле даты конца абонемента;

Класс ABONEMENTREQUEST:

* IDREQUEST – поле уникального идентификатора запроса;
* IDCLIENT – поле уникального идентификатора клиента;
* IDABONEMENT – поле уникального идентификатора абонемента;
* TYPEOFTRAINING – поле тип тренировки;
* TYPEOFSERVICE – поле тип услуги;
* DATAREQUEST – поле даты запроса;

Класс CLIENTSERVICE:

* client: Client – объект класса CLIENT;
* AddClient(client: Client): void – добавление клиента
* GetClient(): List<Client> - получение клиента
* RemoveClient(client: Client): void – удаление клиента
* UpdateClient(client: Client): void – обновление клиента
* GetClientSale(idClient: int): List<Sale> - получение покупки клиента
* GetClientRequest(idClient: int): List<AbonementRequest> - получение запроса клиента;

Класс EMPLOYEESERVICE:

* employee: Employee – объект класса EMPLOYEE;
* Approve(abonementRequest:AbonementRequest): void – подтверждение запроса абонемента
* Deny(abonementRequest:AbonementRequest): void – отклонение запроса абонемента
* AddEmployee(employee: Employee): void – добавление сотрудника
* GetEmployee(): List<Employee> - получение сотрудника
* RemoveEmployee(employee: Employee): void – удаление сотрудника
* UpdateEmployee(employee: Employee): void – обновление сотрудника

Класс LOGPASSERVICE:

* logPas: LogPas– объект класса LOGPAS;
* AddLogPas(logPas: LogPas): void – добавление логина
* GetLogPas(): List<LogPas> - получение логина
* RemoveLogPas(logPas: LogPas):void – удаление логина
* UpdateLogPas(logPas: LogPas): void – изменение логина

Интерфейс IUSERSERVICE:

* GetAbonements(): List<Abonement> - получение абонементов
* Login (logPas: LogPas): void – поле логина
* Registration (logPas: LogPas): void – поле регистрации

Класс ABONEMENTSERVICE:

* abonement: Abonement – объект класса ABONEMENT;
* AddAbonement(abonement:Abonement): void– добавление абонемента
* GetAbonement(): List<Abonement> - получение абонемента
* UpdateAbonement(abonement: Abonement): void – изменение абонемента
* RemoveAbonement(abonement: Abonement): void – удаление абонемента

Класс SALEOFABONEMENTSERVICE:

* sale: SaleOfAbonement– объект класса SALE;
* AddSale(sale: SaleOfAbonement): void – добавление продажи
* RemoveSale(sale: SaleOfAbonement): void – удаление продажи
* UpdateSale(sale: SaleOfAbonement): void – изменение продажи
* GetSale(): List<SaleOfAbonement> - получение продажи

Класс ABONEMENTREQUESTSERVICE:

* request: AbonementRequest– объект класса REQUEST;
* AddRequest(request: AbonementRequest): void– добавление запроса
* GetSale(): List<AbonementRequest>– получение продажи
* RemoveSale(request: AbonementRequest): void– удаление продажи
* UpdateSale(request: AbonementRequest): void- изменение продажи

На рисунке 1 представлена диаграмма классов:

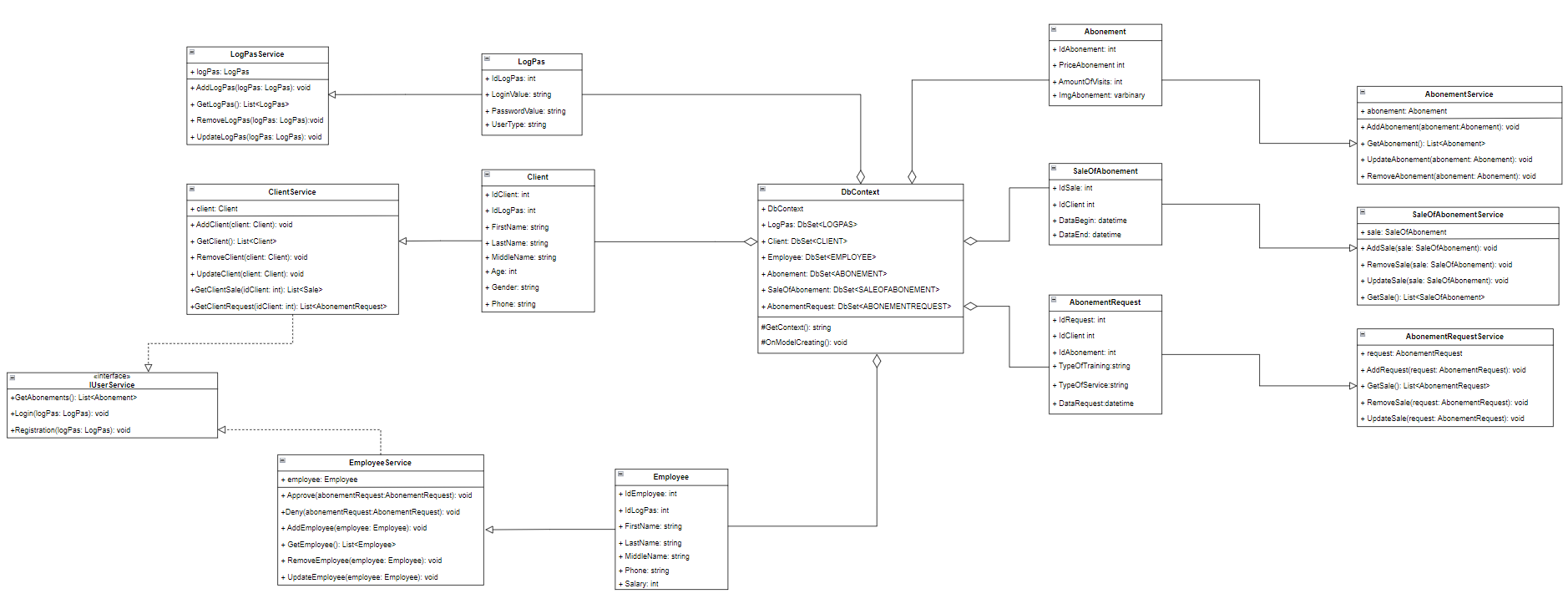


Рисунок 1 – Диаграмма классов

На рисунках 2 и 3 представлена диаграмма пакетов в двух видах:

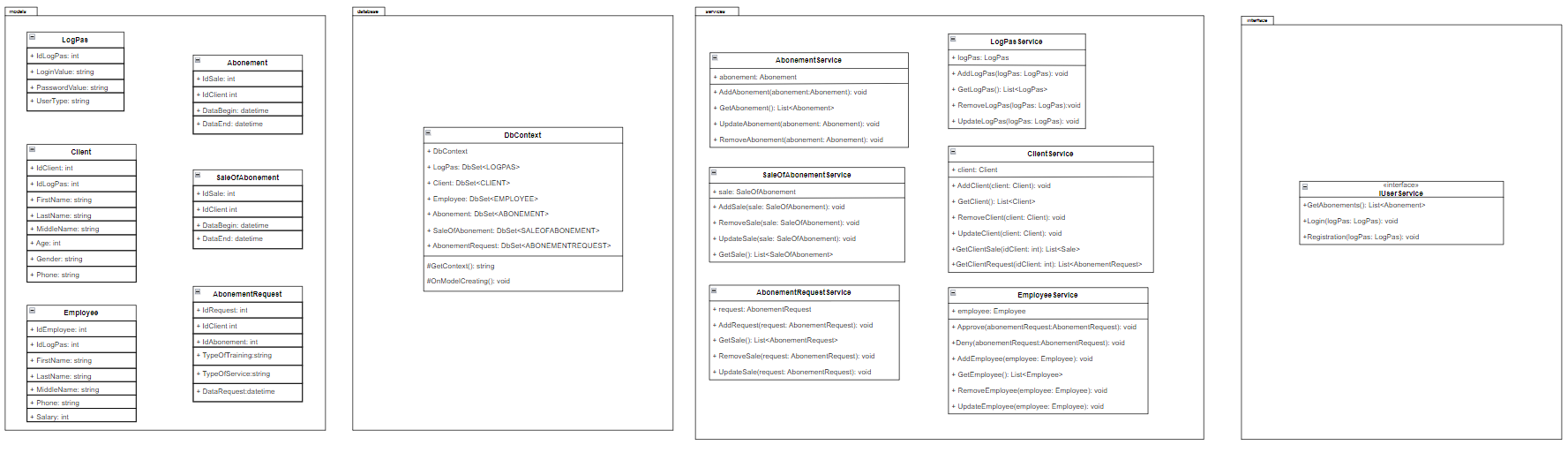


Рисунок 2 – Диаграмма пакетов

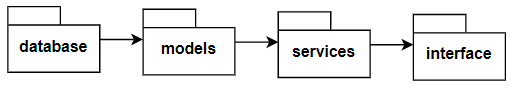


Рисунок 3 – Диаграмма пакетов