**ДОКУМЕНТАЦИЯ ПО СИСТЕМЕ**

**«Human resource manager»**

**ОГЛАВЛЕНИЕ**

[АРХИТЕКТУРА СИСТЕМЫ 3](#_Toc35707817)

[ОПИСАНИЕ БАЗЫ ДАННЫХ 4](#_Toc35707818)

[ОПИСАНИЕ ПРОЕКТА DataAccessLayer 6](#_Toc35707819)

[ОПИСАНИЕ ПРОЕКТА BusinessLogicLayer 8](#_Toc35707820)

[ОПИСАНИЕ ПРОЕКТА WebAPI 10](#_Toc35707821)

[ОПИСАНИЕ ПРОЕКТА PresentationLayer 11](#_Toc35707822)

[ОПИСАНИЕ ПРОЕКТА CommonClasses 12](#_Toc35707823)

[ОПИСАНИЕ ПРОЕКТА ExceptionClasses 13](#_Toc35707824)

# **АРХИТЕКТУРА СИСТЕМЫ**

Вся система состоит из 6 проектов:

*3 основных слоя системы:*

1. DataAccessLayer (отвечает за уровень доступа к базе данных и все операции связанные с базой данных)
2. BusinessLogicLayer (отвечает за обработку логики в приложении и связывает WebAPI и DAL уровни)
3. WebAPI (предоставляет RESTAPI для взаимодействия с логикой приложения)

*Слой для предоставления интерфейса пользователя:*

1. PresentationLayer (отвечает за GUI для приложения и взаимодействует с приложением через WebAPI)

*Вспомогательные библиотеки:*

1. CommonClasses (общие классы для всех слоев системы)
2. ExceptionClasses (классы отвечающие за обработку и логирование ошибок)

# **ОПИСАНИЕ БАЗЫ ДАННЫХ**

База данных состоит из 5 таблиц:

1. ActivityType;
2. Company;
3. Employee;
4. LegalForm;
5. Position;

И хранимых процедур:

* spAddActivityType (добавляет запись в таблицу ActivityType);
* spAddCompany (добавляет запись в таблицу Company);
* spAddEmployee (добавляет запись в таблицу Employee);
* spAddLegalForm (добавляет запись в таблицу LegalForm);
* spAddPosition (добавляет запись в таблицу Position);
* spDeleteActivityType (удаляет по id запись из таблицы ActivityType);
* spDeleteCompany (удаляет по id запись из таблицы Company);
* spDeleteEmployee (удаляет по id запись из таблицы Employee);
* spDeleteLegalForm (удаляет по id запись из таблицы LegalForm);
* spDeletePosition (удаляет по id запись из таблицы Position);
* spGetActivityType (возвращает запись по id из таблицы ActivityType);
* spGetCompany (возвращает запись по id из таблицы Company);
* spGetCompanySize (возвращает количество сотрудников компании);
* spGetEmployee (возвращает запись по id из таблицы Employee);
* spGetLegalForm (возвращает запись по id из таблицы LegalForm);
* spGetPosition (возвращает запись по id из таблицы Position);
* spGetAllActivityTypes (возвращает все записи из таблицы ActivityType);
* spGetAllCompanies (возвращает все записи из таблицы Companies);
* spGetAllEmployees (возвращает все записи из таблицы Employees);
* spGetAllLegalForms (возвращает все записи из таблицы LegalForms);
* spGetAllPositions (возвращает все записи из таблицы Positions);
* spExistsActivityType (проверяет существование записи в бд по id в таблице ActivityType);
* spExistsCompany (проверяет существование записи в бд по id в таблице Company);
* spExistsEmployee (проверяет существование записи в бд по id в таблице Employee);
* spExistLegalForm (проверяет существование записи в бд по id в таблице LegalForm);
* spExistPosition (проверяет существование записи в бд по id в таблице Position);
* spUpdateActivityType (обновляет запись по id из таблицы ActivityType);
* spUpdateCompany (обновляет запись по id из таблицы Company);
* spUpdateEmployee (обновляет запись по id из таблицы Employee);
* spUpdateLegalForm (обновляет запись по id из таблицы LegalForm);
* spUpdatePosition (обновляет запись по id из таблицы Position);

# **ОПИСАНИЕ ПРОЕКТА DataAccessLayer**

Проект отвечает за доступ к данным и операции над ними.

*В нем есть 5 разделов:*

1. DataAccess – отвечает за классы, которые выполняют запросы к базе данных:
   * Класс DataBaseConnection отвечает за подключение к базе данных. Строка подключения берется из файла WebAPI\bin\Debug\netcoreapp2.2\connectionStrings.json, который находится в проекте WebAPI;
   * Класс StoredProcedure за выполнение хранимых процедур и реализует интрефейс IDataBaseCommandExecutor;
2. DataContext – отвечает за классы, которые инкапсулируют выполнение запросов к базе данных и предоставляет интерфейс для работы с данными:
   * ActivityTypeDataAccessLayer;
   * CompanyDataAccessLayer – реализует ICompanyDataAccessLayer ;
   * EmployeeDataAccessLayer;
   * GeneralDataAccessLayer - представляет общую реализацию классов DataAccessLayer и является абстрактным;
   * HrmContext - реализует паттерн UnitOfWork, который предоставляет доступ к классам DataAccessLayer и реализует интерфейс IHrmContext;
   * LegalFormDataAccessLayer;
   * PositionDataAccessLayer;

Классы DataAccessLayer реализуют CRUD операции над сущностями раздела Entities и реализуют интерфейс IDataAccessLayer и наследуются от GeneralDataAccessLayer;

1. Entities – отвечает за сущности базы данных:
   * ActivityType;
   * Company;
   * Employee;
   * LegalForm;
   * Position;
2. Interfaces – содержит интерфейсы проекта DataAccessLayer:
   * ICompanyDataAccessLayer – наследуется от IDataAccessLayer;
   * ICompanyRepository – наследуется от IRepository;
   * IDataAccessLayer;
   * IDataBaseCommandExecutor;
   * IHrmContext;
   * IRepository;
   * IUnitOfWork;
3. Repositories – содержит репозитории (для увеличения гибкости подключения к базе данных):
   * ActivityTypeRepository;
   * CompanyRepository – реализует ICompanyRepository;
   * DataBaseUnitOfWork - реализует паттерн UnitOfWork, который предоставляет доступ к классам Repository и реализует интерфейс IUnitOfWork;
   * EmployeeRepository;
   * GeneralRepository - представляет общую реализацию классов Repository;
   * LegalFormRepository;
   * PositionRepository;

Классы Repository предоставляют доступ к операциям над сущностями раздела Entities и реализуют интерфейс IRepository, наследуются от GeneralRepository.

# **ОПИСАНИЕ ПРОЕКТА BusinessLogicLayer**

Проект отвечает за выполнение логических операции приложения и связывает уровни WebAPI и DataAccessLayer, тем самым скрывая реализацию доступа к данным от WebAPI.

*В нем есть 4 раздела:*

1. DataTransferObject – отвечает за буферные сущности, которые скрывают реализацию уровня DAL:
   * ActivityTypeDTO;
   * CompanyDTO;
   * EmployeeDTO;
   * LegalFormDTO;
   * PositionDTO;

Все классы DTO реализуют интерфейс IDataTransferObject.

1. Converters – отвечает за классы, которые конвертируют сущности уровня DAL в буферные сущности:
   * ActivityTypeConverter;
   * CompanyConverter;
   * EmployeeConverter;
   * GeneralConverter – отвечает за общую реализацию классов Converter;
   * LegalFormConverter;
   * PositionConverter;

Все классы Converter реализуют интерфейс IConverter (кроме GeneralConverter);

1. Interfaces – отвечает за интерфейсы проекта:
   * IConverter;
   * IDataTransferObject;
   * IService;
   * IServiceUnitOfWork;
2. Services – отвечает за сервисы проекта. Они предоставляют доступ к операциям на данными:
   * ActivityTypeService;
   * CompanyService;
   * EmployeeService;
   * GeneralService – отвечает за общую реализацию сервисов и является абстрактным классов;
   * LegalFormService;
   * PositionService;
   * ServiceProvider – реализует паттерн UnitOfWork и интерфейс IServiceUnitOfWork, и предоставляет доступ к сервисам;

Все сервисы реализуют интерфейс IService и наследуются от GeneralService;

# **ОПИСАНИЕ ПРОЕКТА WebAPI**

Проект предоставляет RESTAPI для взаимодействия с логикой приложения.

Содержит только 1 раздел Controllers:

* ActivityTypesController;
* CompaniesController;
* EmployeesController;
* GeneralController – отвечает за общую реализацию контроллеров и является абстрактным;
* LegalFormsController;
* PositionsController;

Все контроллеры наследуются от класса GeneralController.

Все зависимости внедряются в классе Startup.

# **ОПИСАНИЕ ПРОЕКТА PresentationLayer**

Проект отвечает за GUI для приложения и взаимодействует с приложением через WebAPI.

*В нем есть 4 раздела:*

1. Controllers – отвечает за контроллеры приложения, которые обрабатывают запросы прользователей:
   * ActivityTypesController;
   * CompaniesController;
   * EmployeesController;
   * GeneralController – отвечает за общий функционал контроллеров и является абстрактным классом;
   * HomeController;
   * LegalFormsController;
   * PositionsController;

Все контроллеры наследуются от GeneralController.

1. Models – отвечает за сущности уровня PresentationLayer:
   * ActivityTypeModel;
   * CompanyModel;
   * EmployeeModel;
   * ErrorViewModel – отвечает за сущность ошибка для представления Error;
   * LegalFormModel;
   * PositionModel;
2. ViewModels – модели для представлений:
   * ActivityTypeViewModel;
   * CompanyViewModel;
   * EmployeeViewModel;
   * LegalFormViewModel;
   * PositionViewModel;
3. Views – отвечает за представления.

Все зависимости внедряются в классе Startup.

# **ОПИСАНИЕ ПРОЕКТА CommonClasses**

Проект отвечает за общие классы всей системы.

*Он содержит следующие классы:*

* AutoMapper – класс автоматизирует преобразование сущности базы данных в буферную сущность DTO и является статическим классом;
* DataTableMapper – класс автоматизирует создание сущностей и запросов к базе данных;
* IdType – класс отвечает за идентификатор в базе данных и это поле должно быть у всех сущностей, хранящихся в базе данных. У него есть метод ConvertToDBTypeId() который преобразует идентификатор в тип данных, поддерживаемый базой данных;

# **ОПИСАНИЕ ПРОЕКТА ExceptionClasses**

Проект содержит классы, отвечающие за обработку и логирование ошибок.

В нем есть 2 раздела:

1. Exceptions – отвечает за кастомные типы ошибок. Содержит следующие классы:
   * ClientException – отвечает за ошибку, допущенную пользователем приложения и наследуется от класса System.Exception;
2. ExceptionLogers – содержит классы, логирующие ошибки:
   * ExceptionLoger – отвечает за логирование ошибки и является статическим классом.