МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет ИСТ

Кафедра «Информационные системы»

Дисциплина «Разработка профессиональных приложений»

**КУРСОВАЯ РАБОТА**

Тема Разработка программы «Завод «Иди работать». Исполнитель»

Выполнил студент /Гуторов И.А./

подпись инициалы, фамилия

Курс второй Группа ПИбд-23

Направление/специальность 09.03.04 «Программная инженерия» (профиль

«Программная инженерия»)

Руководитель доцент Эгов Е.Н.

должность, ученая степень, ученое звание фамилия, имя, отчество

Дата сдачи:

« » 2025 г.

Дата защиты:

« » 2025 г. Оценка:

Ульяновск 2025 г

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет ИСТ

Кафедра «Информационные системы»

Дисциплина «Разработка профессиональных приложений»

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

Студенту ПИбд-23 Гуторов И.А.

группа фамилия, инициалы

Тема работы Разработка программы «Завод «Иди работать». Исполнитель»  
Срок сдачи законченного проекта «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

**Исходные данные к работе**: описание задания по теме, утвержденной распоряжением деканата ФИСТ

**Рекомендуемая литература**: курс лекций по дисциплине «Технологии программирования», методические указания к лабораторным работам по дисциплине «Технологии программирования», интернет-источники.

**Содержание пояснительной записки** (перечень подлежащих разработке вопросов)   
Введение. Описание актуальности задачи.

Первая глава. Описание предметной области, поиск аналогов, ТЗ. Представление диаграмм с их описанием.

Вторая глава. Представление руководства пользователя для разработанного проекта.

Третья глава. Представление руководства программиста для разработанного проекта.

**Перечень графического материала** (с точным указанием обязательных чертежей) Диаграммы UML: диаграммы вариантов использования (use-case), диаграмма последовательности (sequence), диаграмма состояния (state-machine).

ER-диаграмма.  
Скриншоты разработанного программного продукта.

Руководитель доцент \_\_\_\_\_\_\_\_\_\_\_\_\_ /Эгов Е.Н./

должность подпись инициалы, фамилия

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   **/** Гуторов И.А.**/**

подпись инициалы, фамилия

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЗЫВ  
руководителя на курсовую работу**

студента Гуторова Ивана Александровича

фамилия, имя и отчество

Факультет ИСТ группа ПИбд-23 курс второй

Дисциплина «Разработка профессиональных приложений»

Тема работы Разработка программы «Завод «Иди работать». Исполнитель»

Отмечаются следующие моменты: актуальность темы исследования; соответствие содержания и структуры курсовой работы ее теме; степень разработанности проблемы, наиболее интересно исследованные вопросы. Оценивается степень самостоятельности и инициативы студента; умение пользоваться различными источниками информации; уровень его теоретической подготовки; умение анализировать научные материалы, делать практические выводы; знание основных концепций, научной и специальной литературы по избранной теме. Содержится оценка проекта (работы) руководителем.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель доцент /Эгов Е.Н./

должность, учёная степень, ученое звание подпись инициалы, фамилия

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

**Введение**

В современном мире, характеризующемся высокими темпами промышленного развития и ужесточением конкуренции, ключевым фактором успеха любого производственного предприятия является эффективное управление своими ресурсами. Одним из наиболее важных и сложных в управлении активов является человеческий капитал, а именно – распределение трудовых ресурсов по задачам в соответствии с их квалификацией, загрузкой и приоритетами производства.

На многих промышленных предприятиях процесс составления графиков смен, распределения заданий и учета рабочего времени до сих пор ведется с помощью электронных таблиц (например, Microsoft Excel) или даже вручную на бумажных носителях. Этот подход является трудоемким, подверженным ошибкам и не позволяет оперативно реагировать на изменяющиеся условия производственного процесса, такие как поломка станка, болезнь сотрудника или поступление срочного заказа.

Целью данной курсовой работы является разработка программы «Завод «Иди работать». Исполнитель», предназначенной для автоматизации процесса назначения производственных заданий сотрудникам цеха.

**Первая Глава**

Предметная область и аналоги

Программа «Завод «Иди работать». Исполнитель» предназначена для ведения учета деталей. Пользователи формируют детали, в каких изделиях они используются, и какие производства будут задействованы в изготовлении. Программа предоставляет возможность получения отчетов по изделиям. Пользователи могут создавать детали. Могут создавать изделия и выбирать к ним детали. Могут создавать производства и выбирать к ним детали. Могут связывать производства с цехами. Могут получать списки цехов по выбранным деталям в формате текстового документа или электронной таблицы. Могут получать сведения на почту или в программе по деталям с расшифровкой по производствам и станкам за период.

ERP-системы для производства:

* SAP S/4HANA Manufacturing

Охватывает управление заготовками, производственными этапами, станками, персоналом. Интегрируется с MES (Manufacturing Execution System) для контроля выполнения планов. Возможность генерации отчетов в различных форматах.

* Oracle ERP Cloud (Manufacturing Module)

Имеет функции управления материалами, оборудованием, цехами; планирования производственных процессов и отслеживания этапов; аналитики по загрузке станков и сотрудников.

* Microsoft Dynamics 365 Supply Chain Management

Реализован учет деталей, заготовок, связь с изделиями; управление цехами и распределение работников. Можно вести гибкую отчетность с экспортом в Excel/PDF.

2. Специализированные MES-системы

* ProShop

Для машиностроительных предприятий. Учет станков, работников, деталей. Отслеживание этапов производства и формирование отчетов.

* Katana MRP

Управление заготовками, материалами, производственными планами. Визуализация загрузки станков и цехов. Интеграция с почтой для уведомлений.

4. Системы для учета персонала

* BambooHR

Учет работников, их привязка к отделам и оборудованию. Анализ загрузки сотрудников.

* 1С:Предприятие

Учет персонала, расчет заработной платы, учет рабочего времени, кадровое планирование. Широкое применение отчетности.

Техническое задание на разработку программного продукта «Завод «Иди работать». Исполнитель»

* Регистрация. Для регистрации пользователь должен заполнить следующие данные:
  + Логин. Уникальный, не длиннее 50 символов, только из латинских букв и цифр.
  + ФИО. Только из кириллических букв, 3 слова через пробел.
  + Дата рождения. В формате ДД.ММ.ГГГГ.
  + Пароль. Из 8 или более символов, содержит буквы из разных регистров, цифры и особые символы.
* Авторизация. Пользователь вводит логин и пароль. Если такой пользователь есть в системе, то происходит переход к основной форме. Если такого пользователя нет в системе, выводится сообщение об этом с просьбой ввести заново данные.
* Основная форма. Состоит из пунктов меню и логотипа. Через пункты меню можно перейти на формы работы с деталями, изделиями и производствами (в том числе и привязка производств), на форму получения списка изделий и на форму получения отчета по изделиям. Дополнительно, если реализовано, то вызов формы рекомендательной подсистемы.
* Формирование детали (CRUD). Имеется форма со списком всех деталей пользователя и кнопки для создания, изменения и удаления детали. По детали хранятся следующие сведения:
  + Id: varchar(100)
  + Название: varchar(100)
  + Материал: varchar(100)
* Формирование изделия (CRUD). Имеется форма со списком всех изделий пользователя и кнопки для создания, изменения и удаления изделия. По изделию хранятся следующие сведения:
  + Id: varchar(100)
  + Название: varchar(100)
  + Дата создания: date

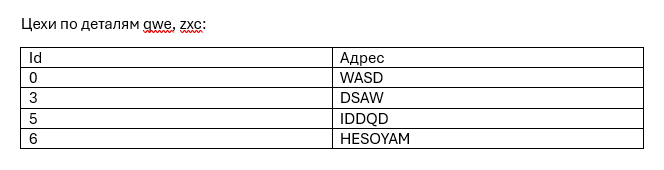
Имеется возможность выбрать в создаваемую запись доступные детали **(из тех, что создавал сам пользователь)** из списка (несколько) и сохранить изменения.

* Формирование производства (CRUD). Имеется форма со списком всех производств пользователя и кнопки для создания, изменения и удаления производства. По производству хранятся следующие сведения:
  + Id: varchar(100)
  + Адрес: varchar(100)

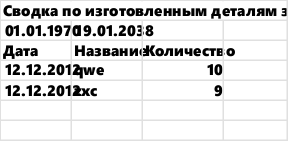
Имеется возможность выбрать в создаваемую запись доступные детали **(из тех, что создавал сам пользователь)** из списка (несколько) и сохранить изменения.

* Привязка производств **(из тех, что создавал сам пользователь)** к цехам. Пользователь может выбрать производство, цех и связать их.
* Получение списка. Пользователь может получить список цехов по выбранным деталям **(из тех, что создавал сам пользователь)**. Пользователь отмечает интересующие его детали, указывает формат файла (doc или xls), указывает имя и место сохранения файла и получает интересующий его список.

При выводе в doc-файл формат документа будет следующим:



При выводе в xls-файл формат документа будет следующим:

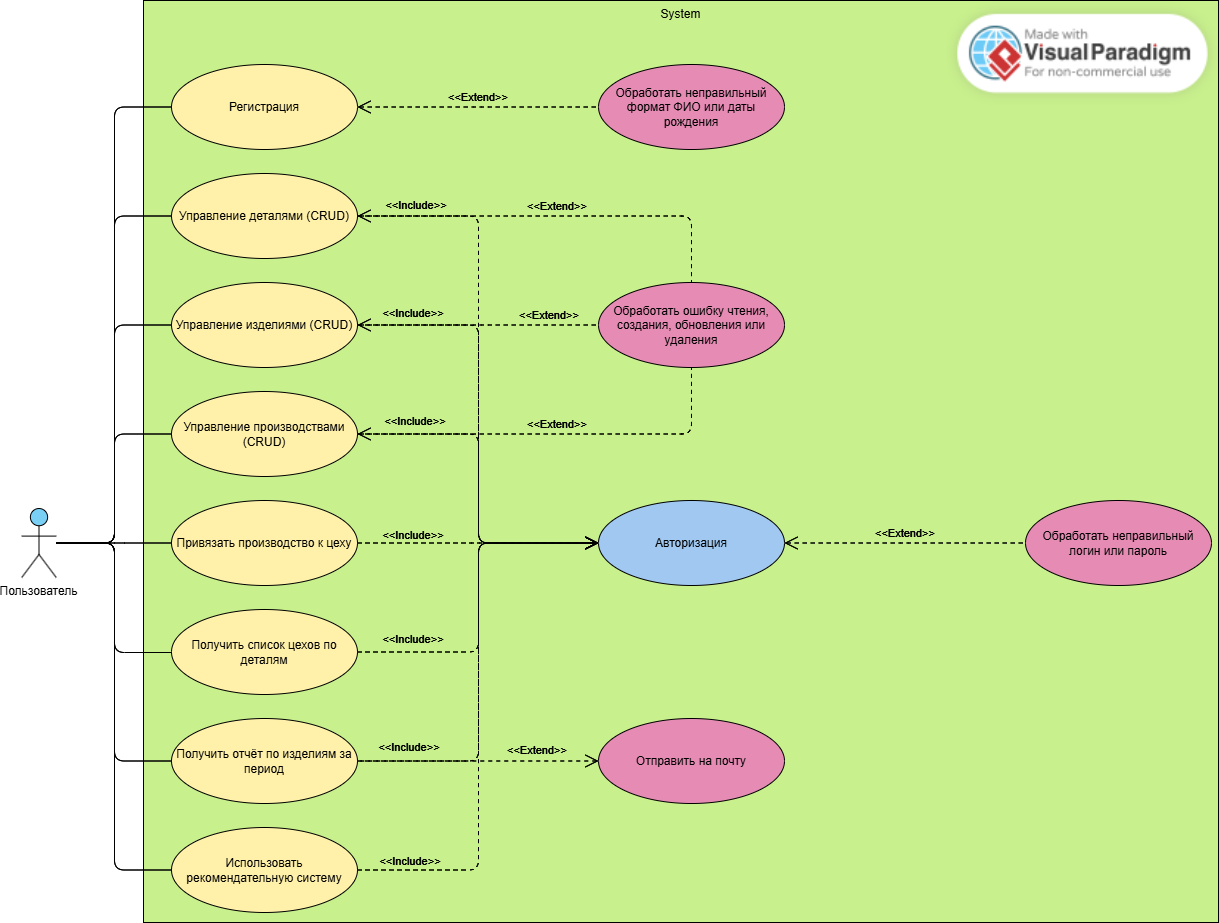


* Пользователь может получить сведения за период по деталям (из тех, что создавал сам пользователь), для которых требовались производства и станки. Для этого он должен указать интересующий его период, выбрать вариант получения сведений (отправка на почту или вывод на форму) и получить отчет.
* Метод создания работников. Метод вызывается при старте программы, проверяет, есть ли работники в базе данных, и, если их нет, рандомно создавать их там.
* Метод создания станков. Метод вызывается при старте программы, проверяет, есть ли станки в базе данных, и, если их нет, рандомно создавать их там.
* Метод создания цехов. Метод вызывается при старте программы, проверяет, есть ли цехи в базе данных, и, если их нет, рандомно создавать их там.

В качестве интерфейса пользователя будет выступать Web-клиент, разработанный на Web API.

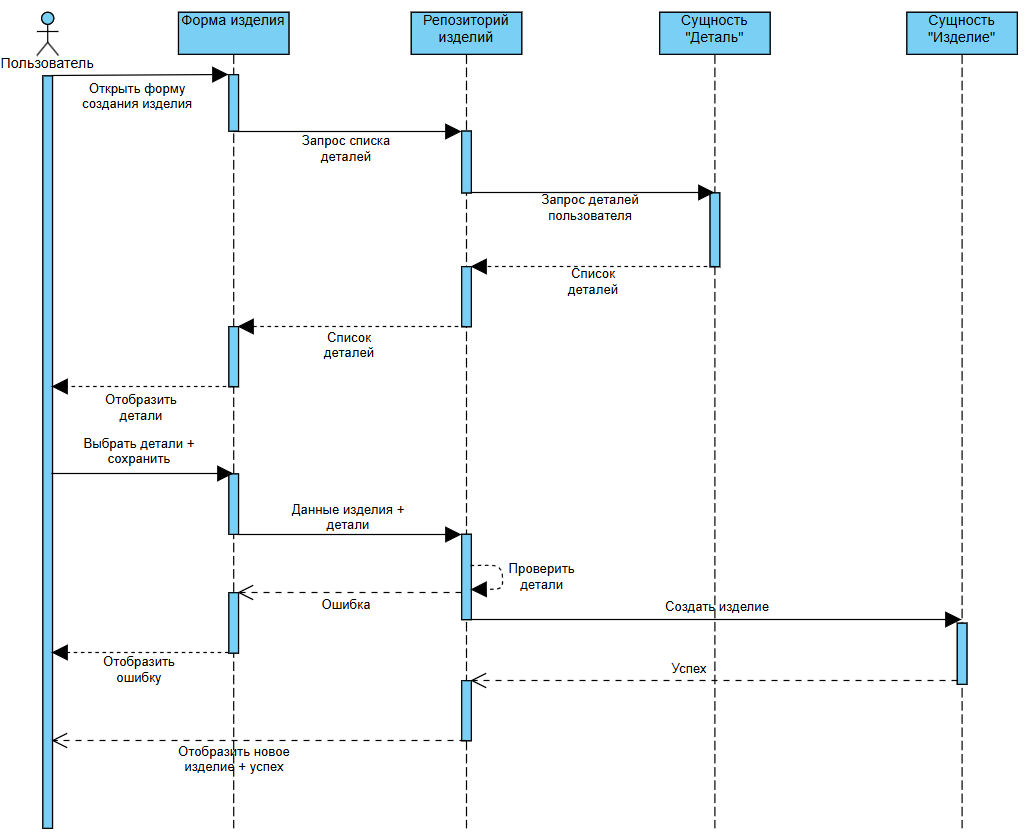
Диаграммы для разработки программного продукта «Завод «Иди работать». Исполнитель»

Диаграмма вариантов использования



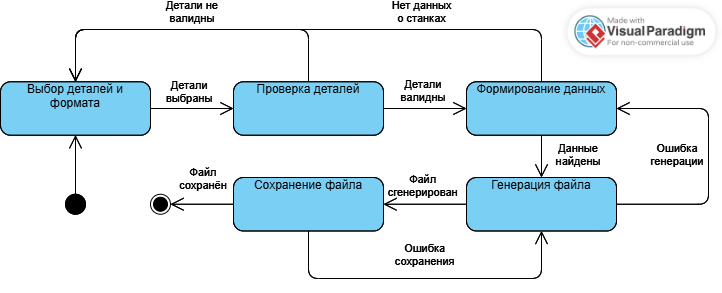
Система включает акторов (пользователей), взаимодействующих с функционалом управления производством. Основные прецеденты: регистрация, CRUD-операции для деталей, изделий и производств, привязка производств к цехам, формирование отчетов за период и использование рекомендательной системы. Связи между акторами и прецедентами прямые: пользователь инициирует действие (например, «Создать изделие»), а система обрабатывает запрос. Отдельно выделены системные прецеденты: авторизация, обработка ошибок ввода (некорректные ФИО, дата рождения), ошибок CRUD-операций и аутентификации (неверный логин/пароль). Каждый прецедент связан с актором через ассоциацию, а зависимости между прецедентами (например, «Проверить детали» перед «Создать изделие») показаны пунктирными стрелками с меткой «include».

Диаграмма последовательности



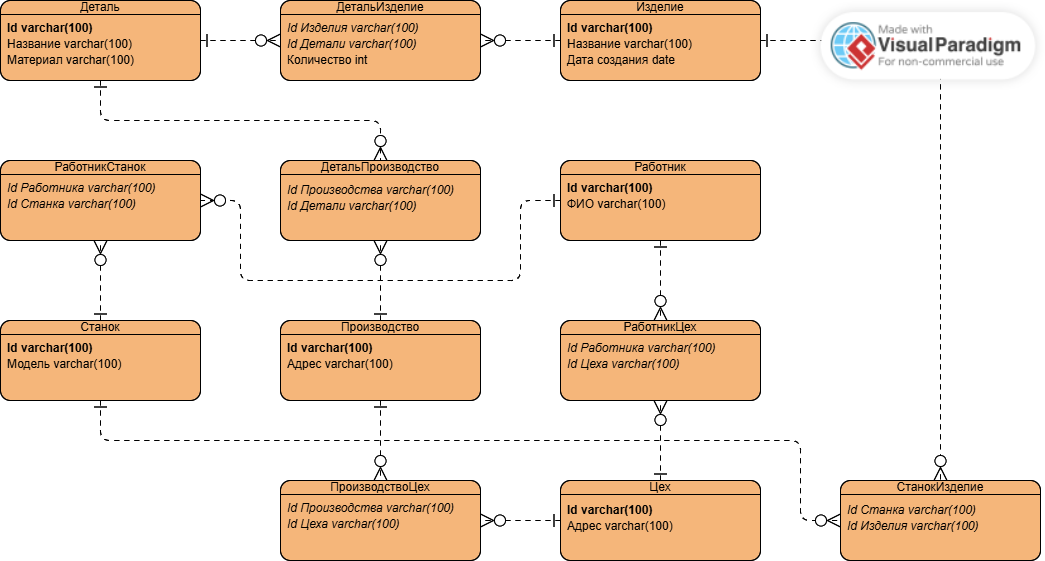
Процесс начинается с открытия формы создания изделия. Пользователь запрашивает список деталей, выбирает их и сохраняет выбор. Система проверяет валидность деталей (наличие данных о станках и материалах). Если данные не валидны — отображается ошибка. При успешной проверке формируются данные изделия, которые сохраняются в БД. Если сохранение завершается ошибкой (например, дубликат Id), процесс прерывается с уведомлением. Успешное создание изделия приводит к отображению его данных и статуса «Успех». Дополнительно предусмотрен этап генерации файла (например, техкарты): после проверки деталей система формирует файл, сохраняет его и обрабатывает ошибки (например, отсутствие прав доступа к директории).

Диаграмма состояний



Процесс начинается с выявления невалидных данных: система проверяет детали и обнаруживает ошибки (например, "Детали не валидны" или "Нет данных о станках"). Пользователь повторно выбирает детали и формат файла ("Выбор деталей и формата"). После подтверждения выбора ("Детали выбраны") система выполняет проверку корректности данных ("Проверка деталей"). Если детали не проходят проверку, процесс возвращается к этапу выбора. При успешной проверке ("Детали валидны") система переходит к формированию данных для файла ("Формирование данных") и запускает его генерацию ("Генерация файла"). Если генерация завершается успешно ("Файл сгенерирован"), система сохраняет файл ("Сохранение файла"). При ошибке сохранения ("Ошибка сохранения") процесс прерывается с уведомлением. Если на этапе генерации возникает проблема ("Ошибка генерации"), система также останавливает процесс. Успешное завершение отмечено статусом "Файл сохранён".

Схема БД для разработки программного продукта «Завод «Иди работать». Исполнитель»



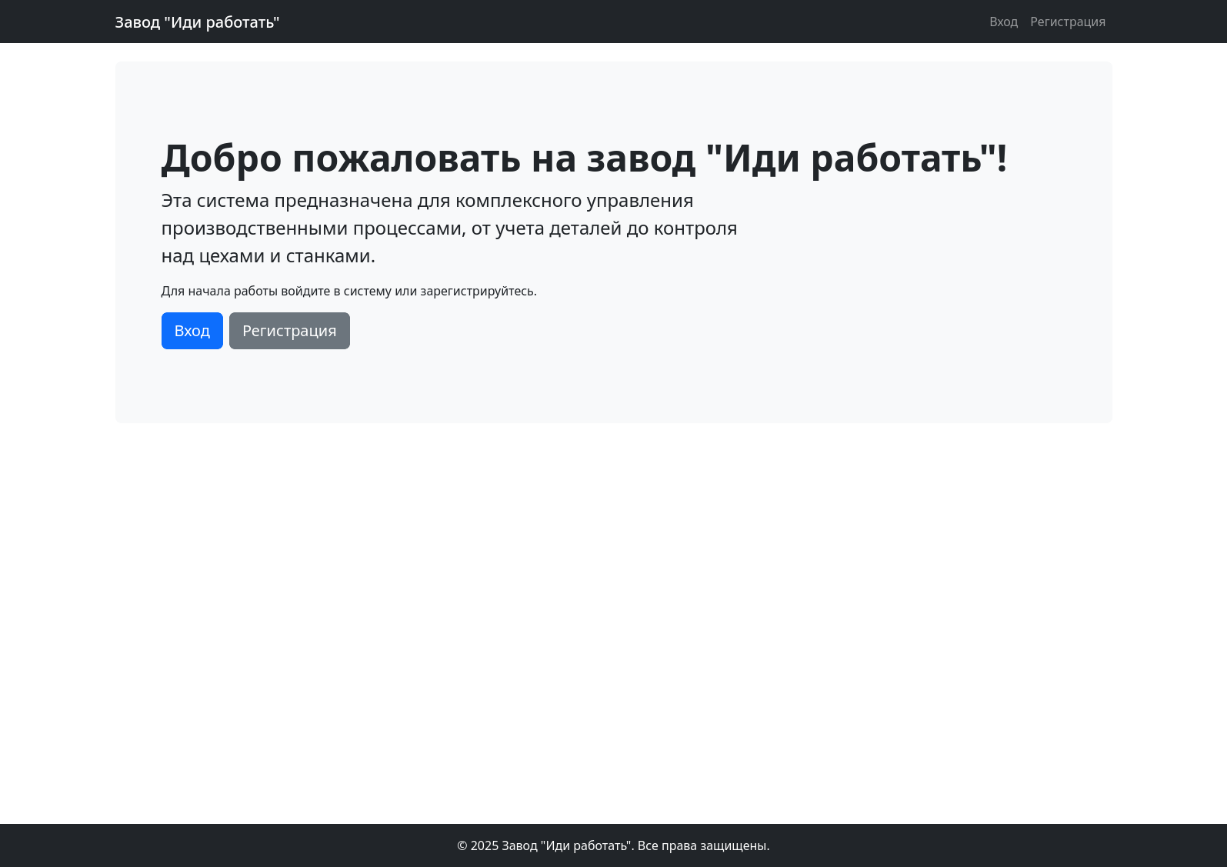
Сущности включают Деталь (атрибуты: Id, название, материал), Изделие (Id, название, дата создания), Производство (Id, адрес), Цех (Id, адрес), Станок (Id, модель), Работник (Id, ФИО). Связи:

* Изделие и Деталь связаны через таблицу Деталь/Задание (отношение многие-ко-многим: одно изделие включает несколько деталей, одна деталь используется в нескольких изделиях).
* Производство и Цех связаны через ПроизводствоЦех (многие-ко-многим: производство может быть привязано к нескольким цехам).
* Работник и Станок связаны через РаботникСтанок (многие-ко-многим: работник обслуживает несколько станков).
* Деталь и Производство связаны через ДетальПроизводство (многие-ко-многим: деталь используется в нескольких производствах).
* Работник и Цех связаны через РаботникЦех (многие-ко-многим: работник может быть прикреплен к нескольким цехам).
* Станок и Изделие связаны через СтанокИзделие (многие-ко-многим: станок участвует в создании нескольких изделий).

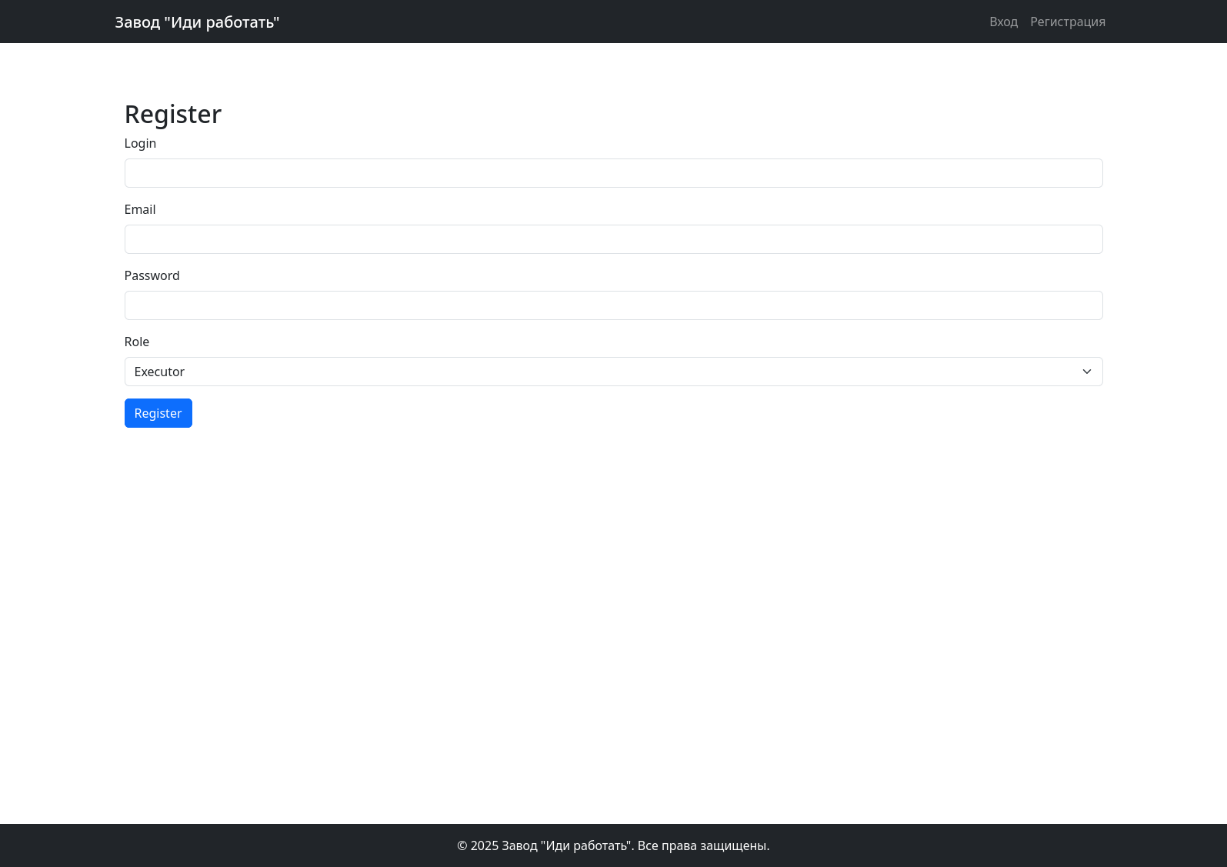
**Вторая глава**

Руководство пользователя на программный продукт «Завод «Иди работать». Исполнитель»

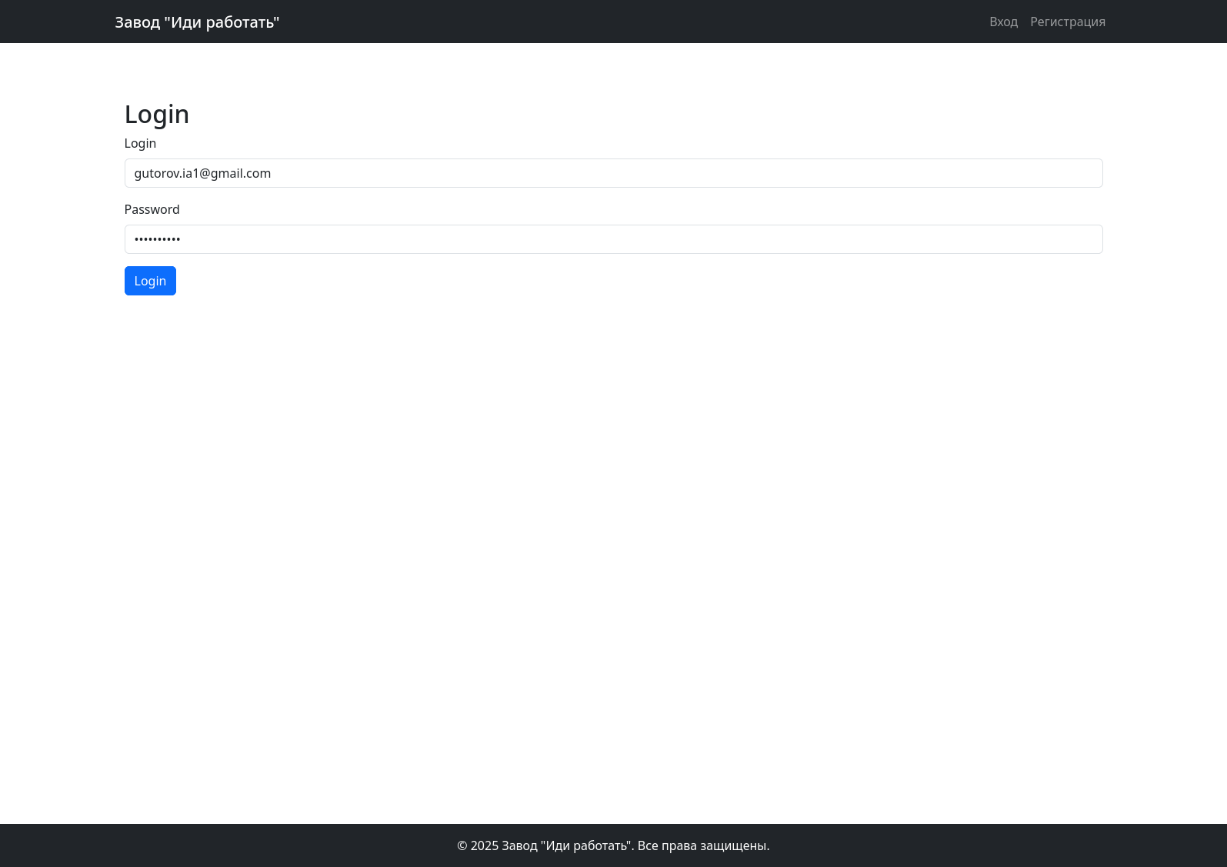
1. При входе на сайт открывается стартовая страница.



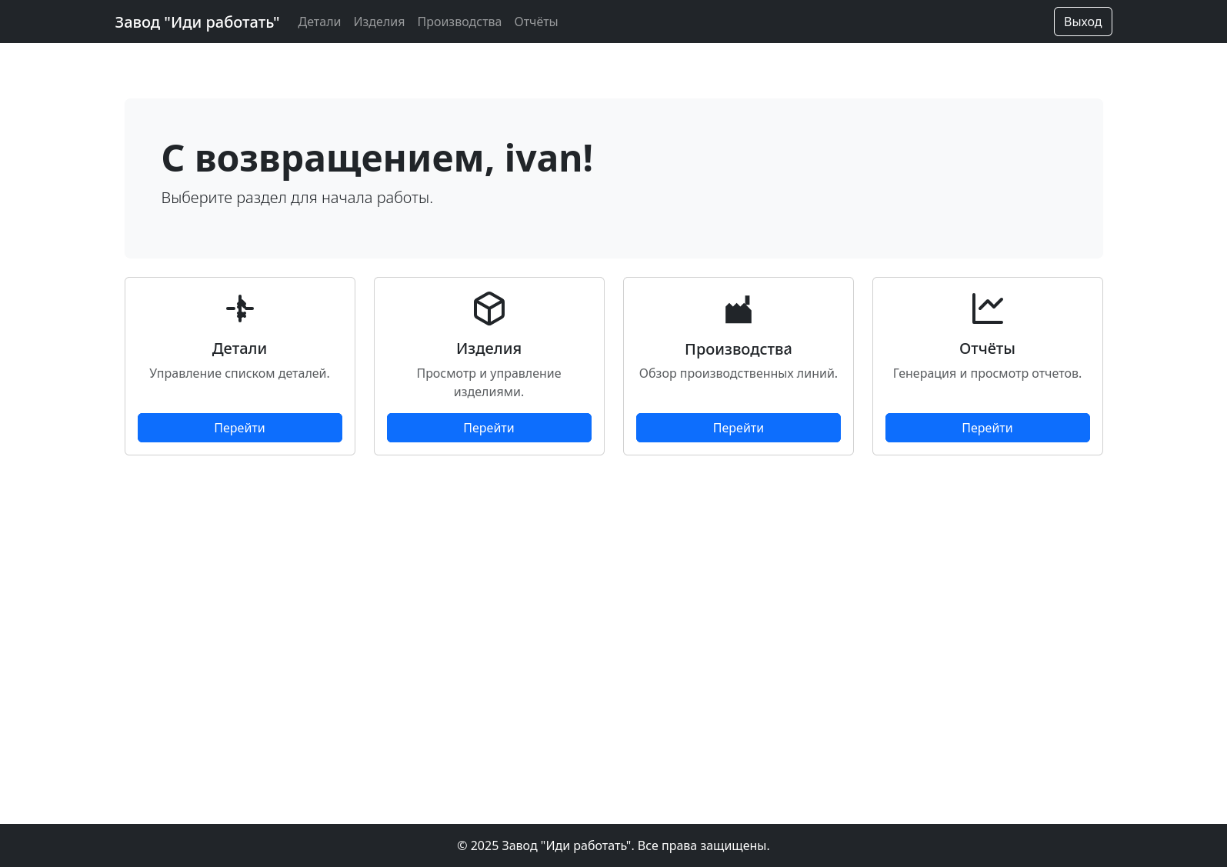
* 1. Если нажать «Регистрация», откроется страница регистрации пользователя.



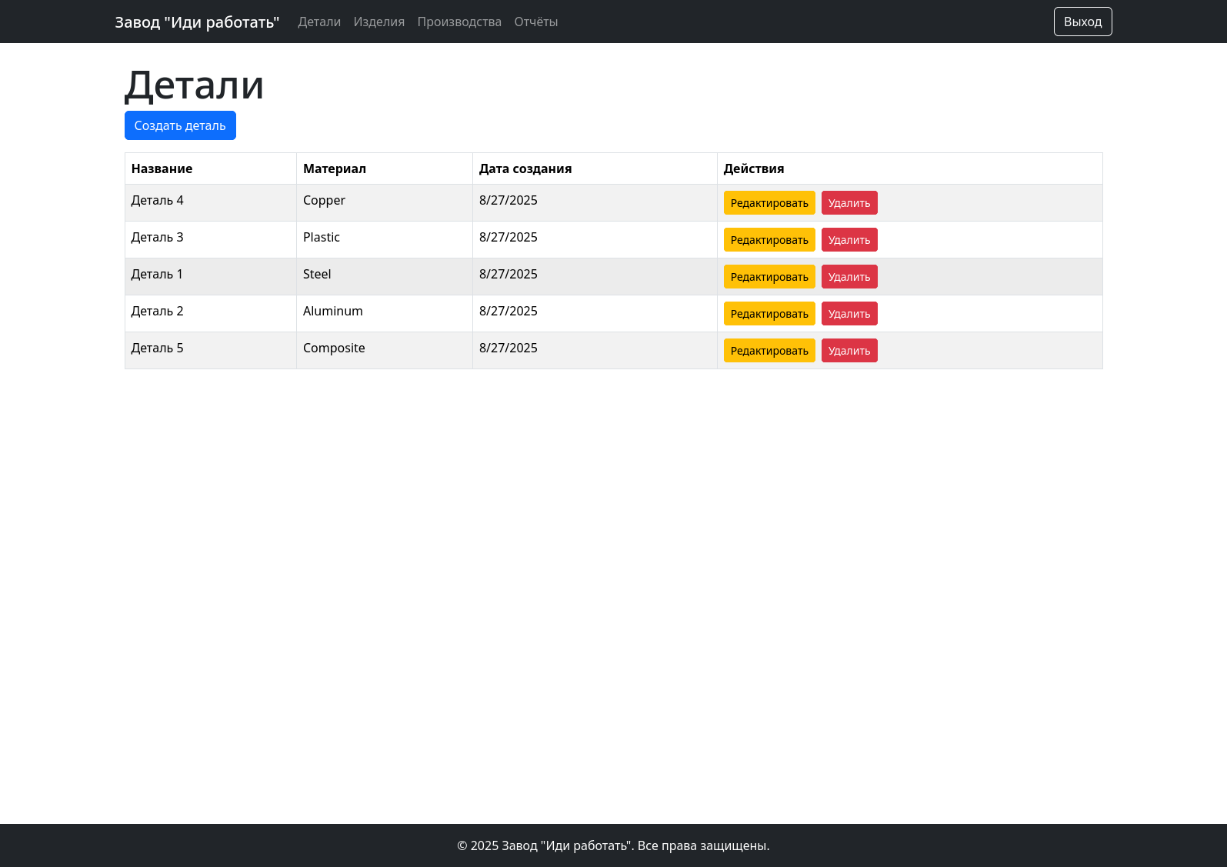
* 1. Если нажать «Вход», откроется страница входа в аккаунт.



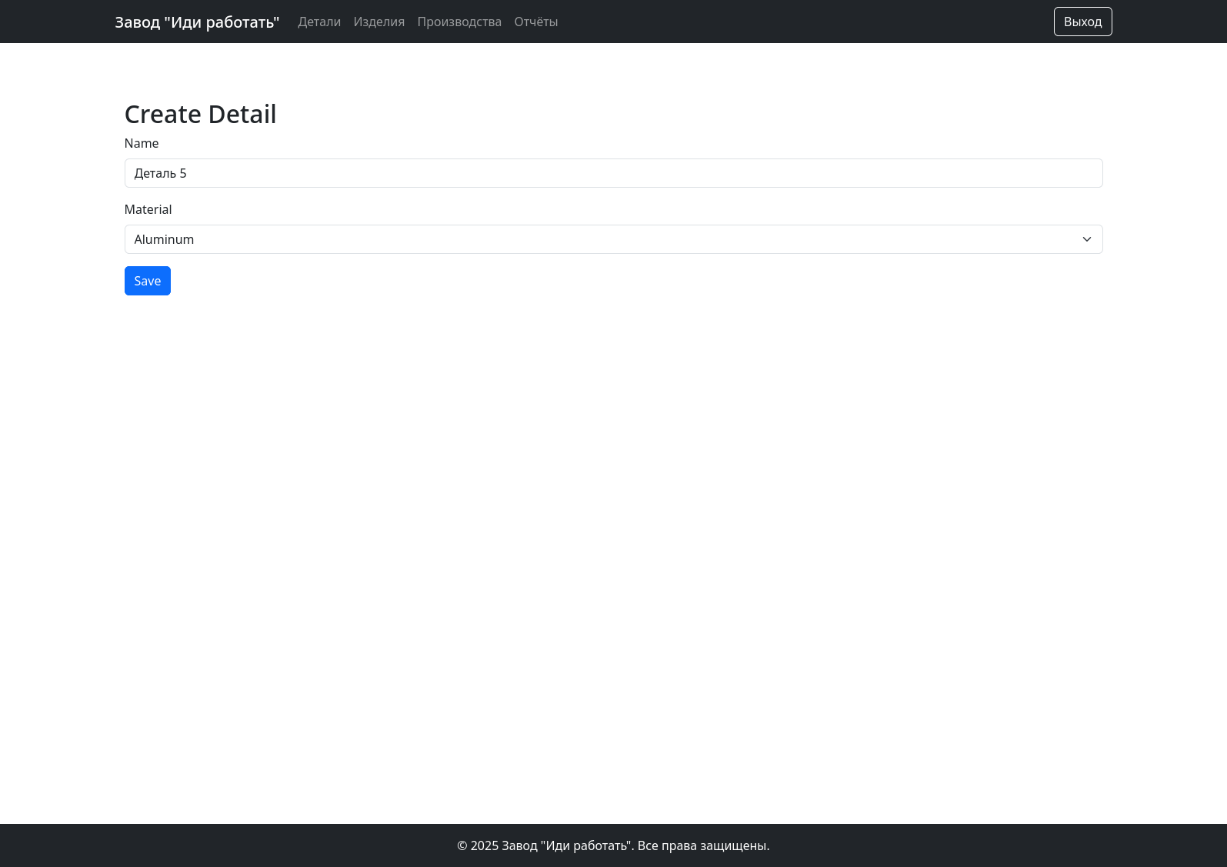
1. После входа в аккаунт откроется главная страница.



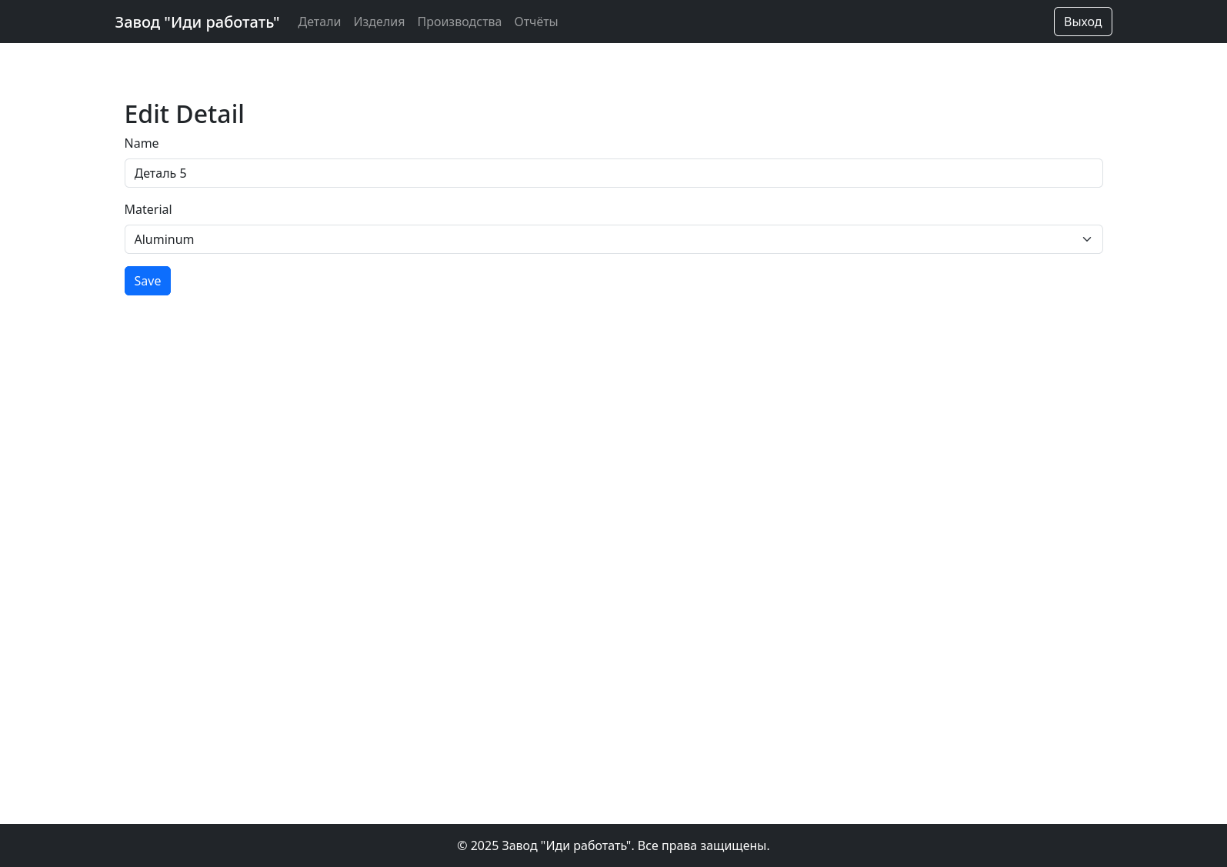
1. После нажатия на кнопку «Детали» откроется страница с деталями.



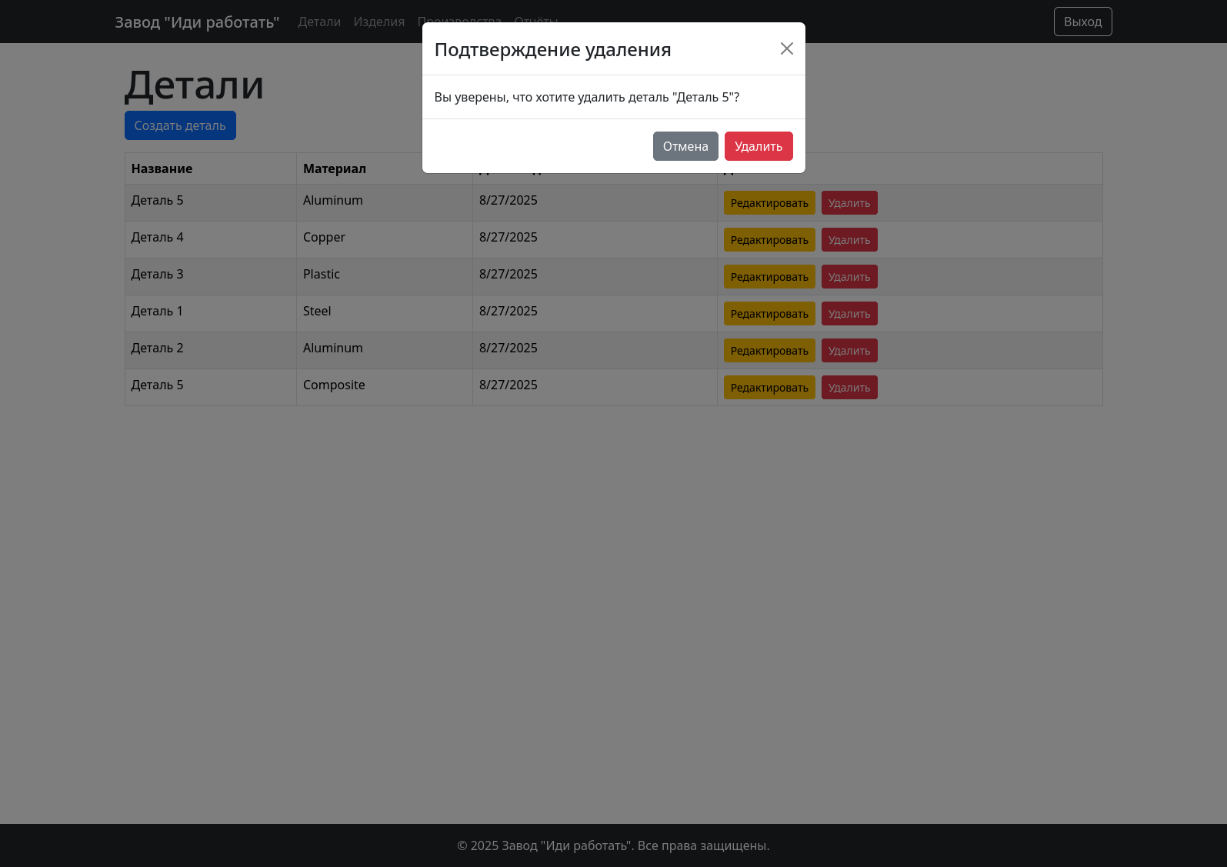
* 1. После нажатия на кнопку «Создать деталь» откроется страница создания детали.



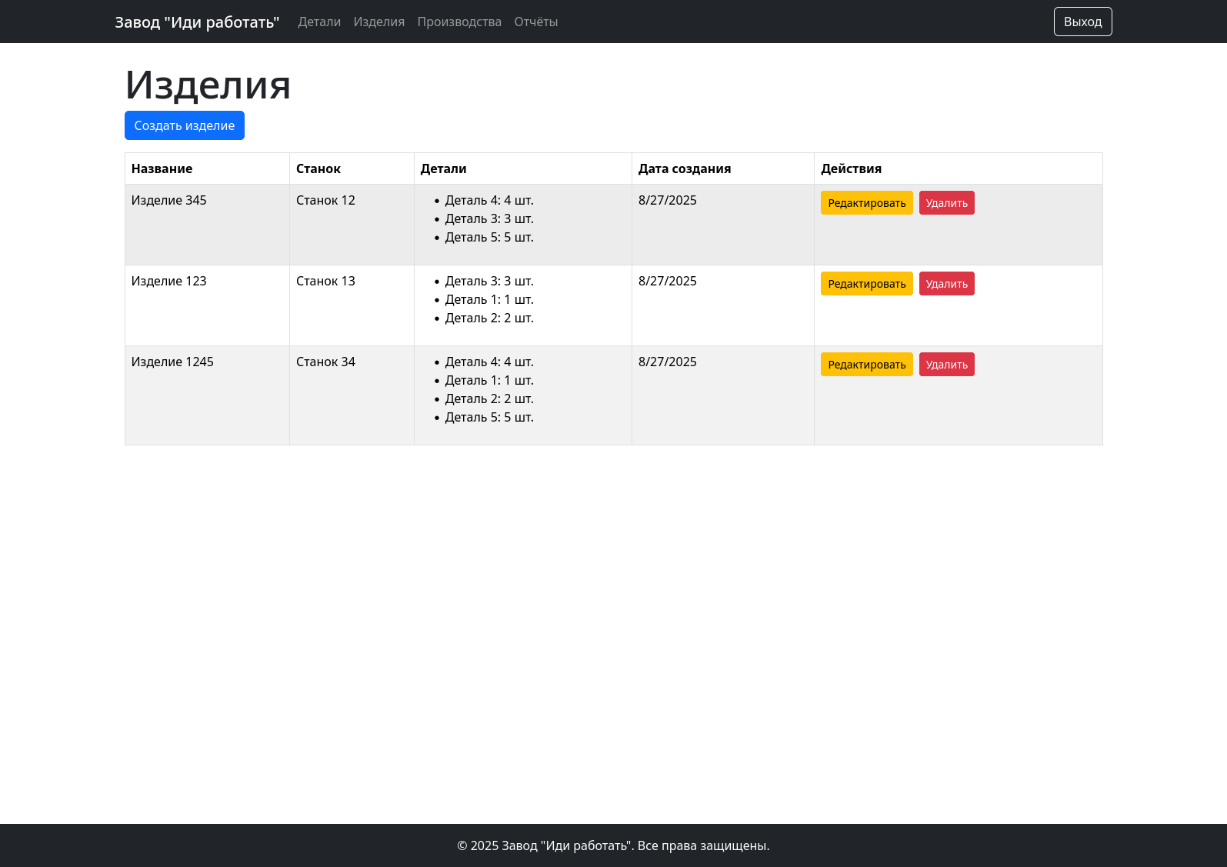
* 1. После нажатия на кнопку «Редактировать» откроется страница редактирования детали.



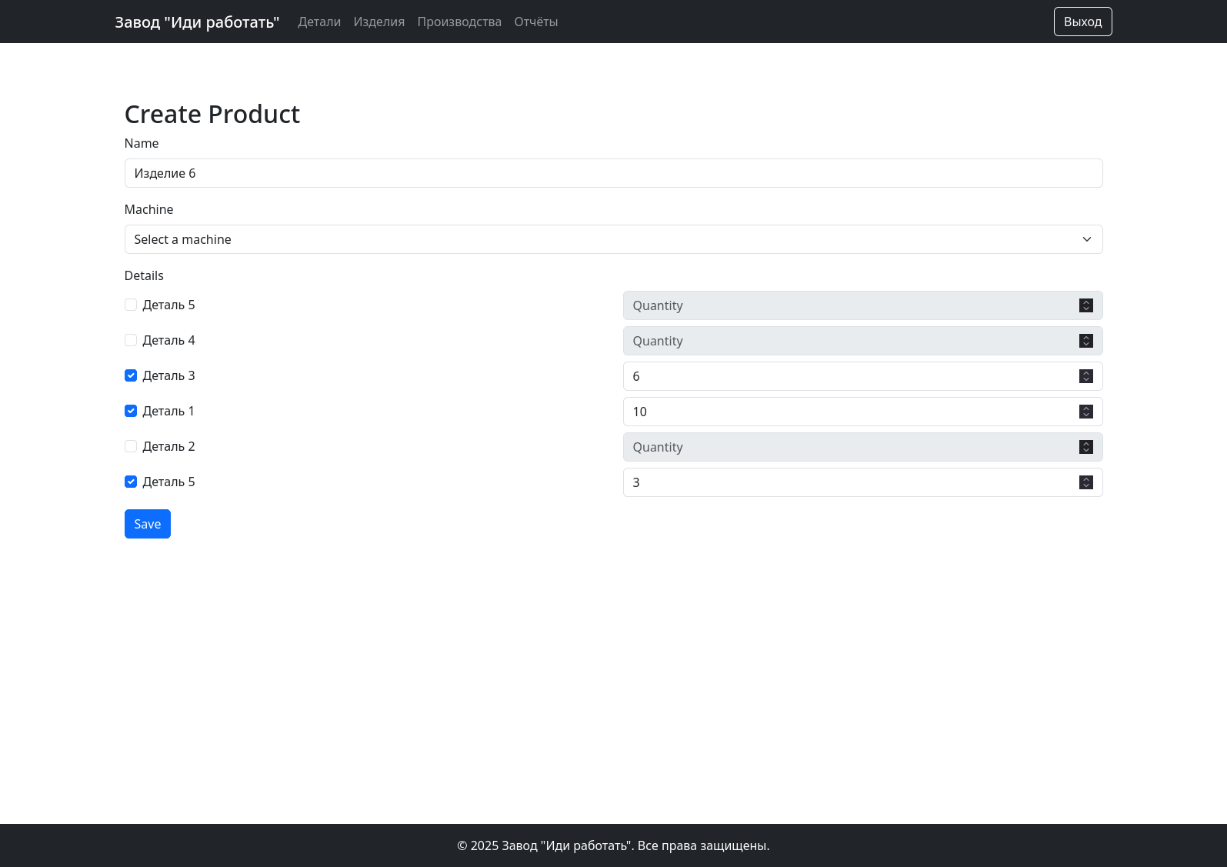
* 1. После нажатия на кнопку «Удалить» на детали всплывёт окно подтверждения удаления детали.



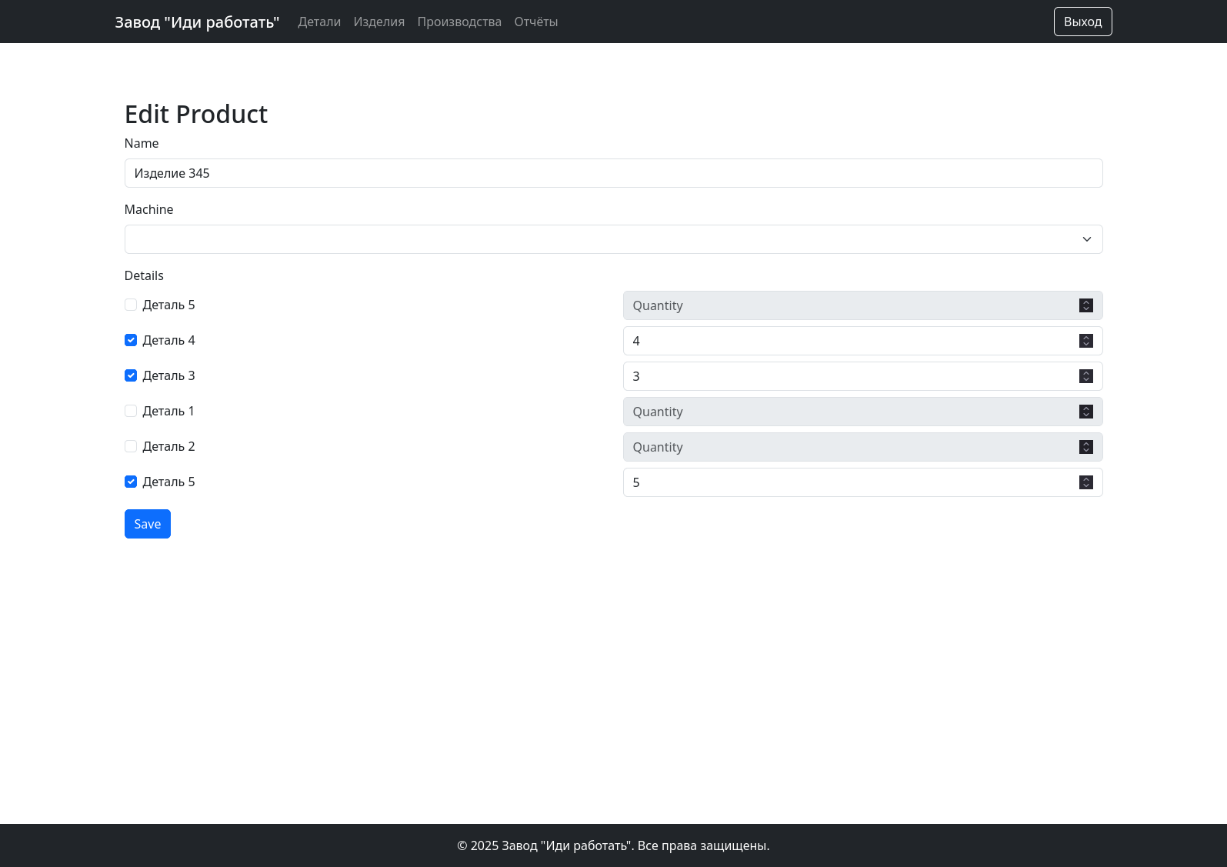
1. После нажатия на кнопку «Изделия» откроется страница с изделиями.



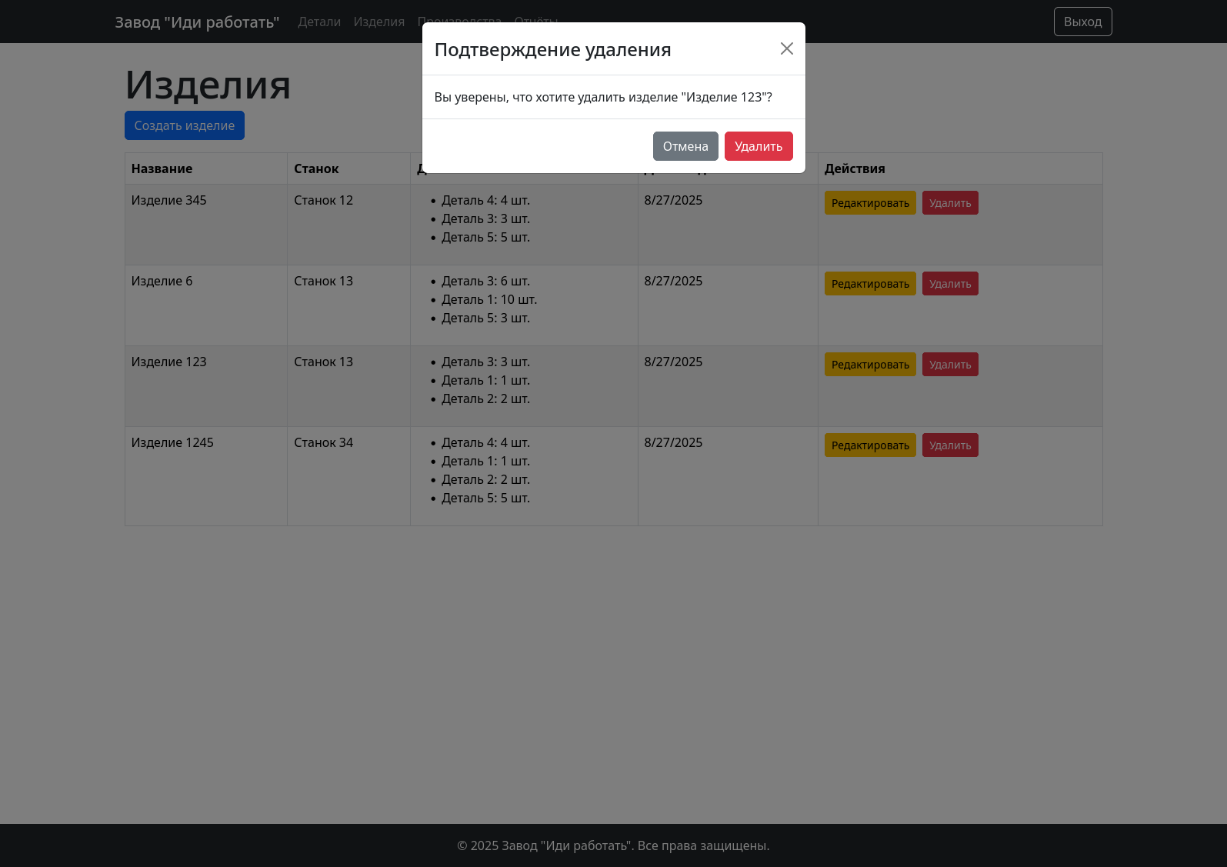
* 1. После нажатия на кнопку «Создать изделие» откроется страница создания изделия.



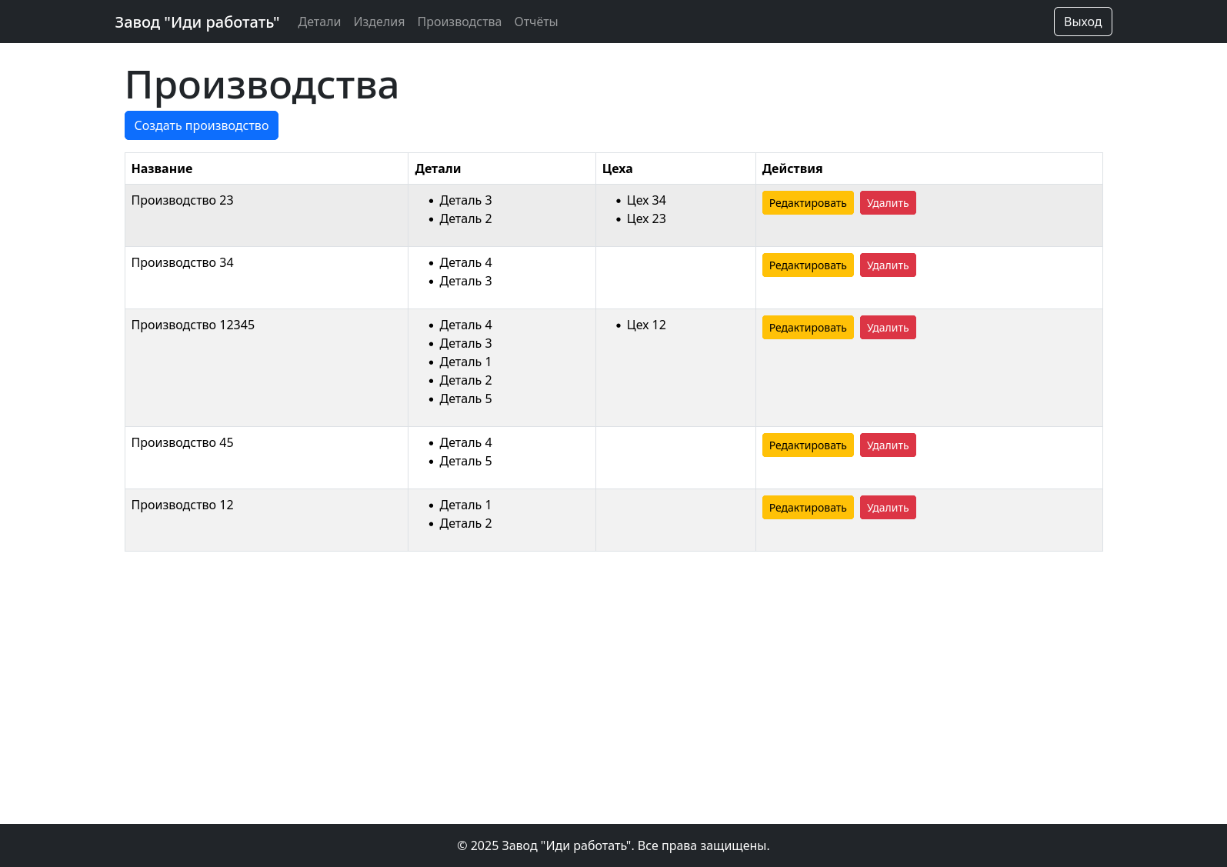
* 1. После нажатия на кнопку «Редактировать» откроется страница редактирования изделия.



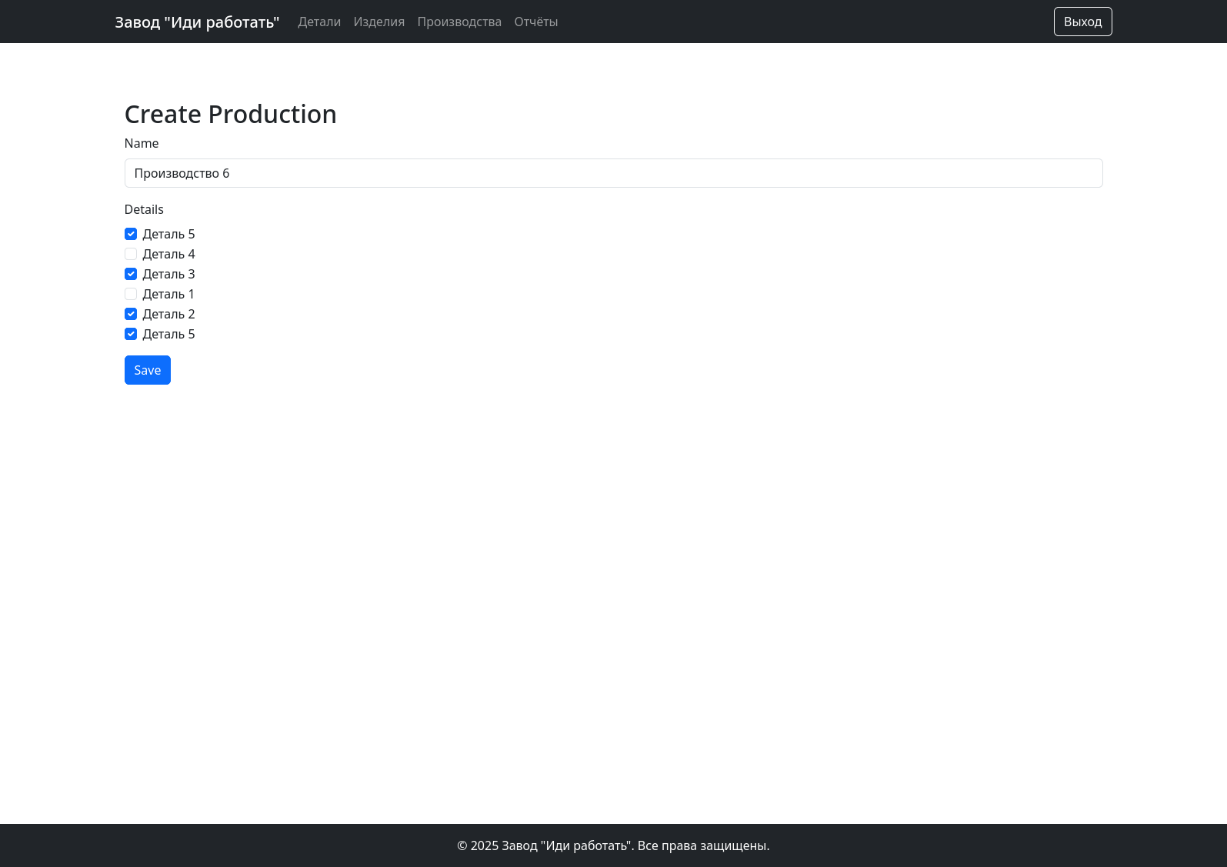
* 1. После нажатия на кнопку «Удалить» на изделии всплывёт окно подтверждения удаления изделия.



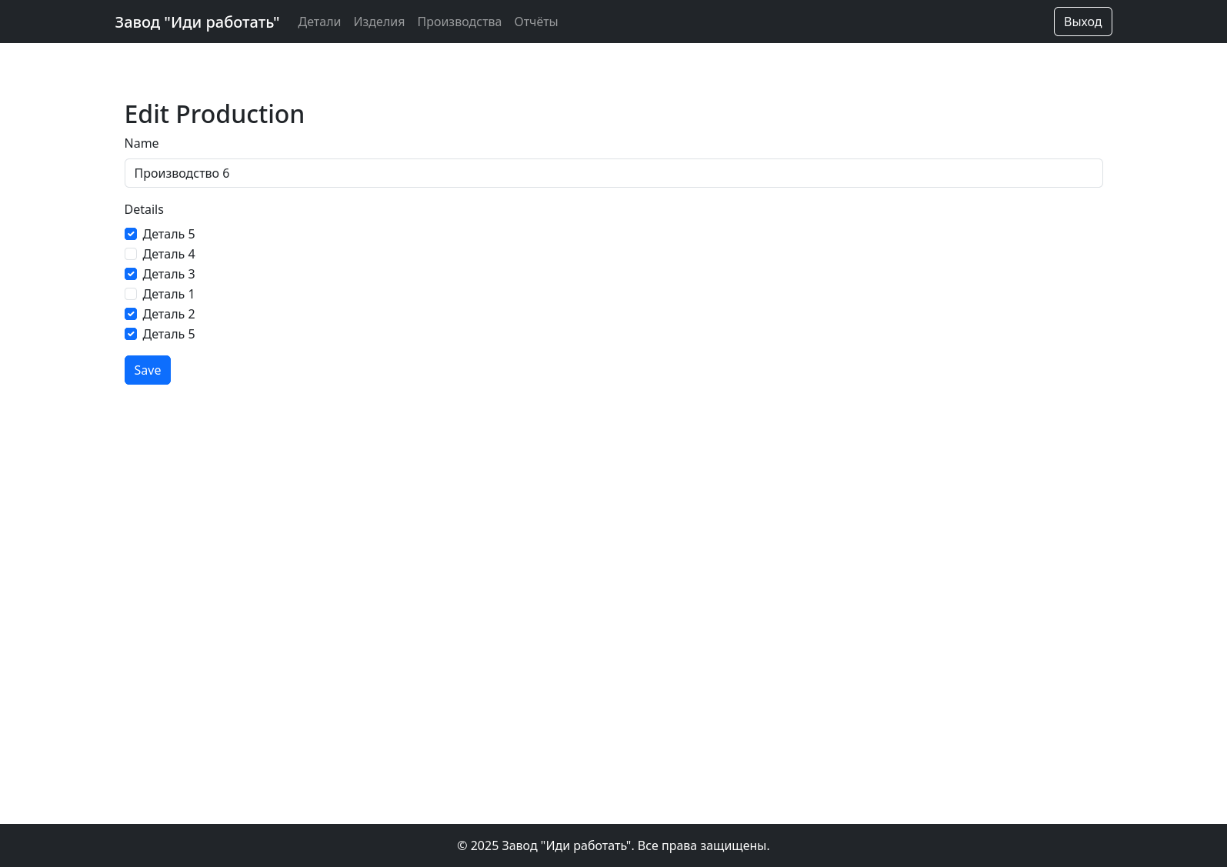
1. После нажатия на кнопку «Производства» откроется страница с производствами.



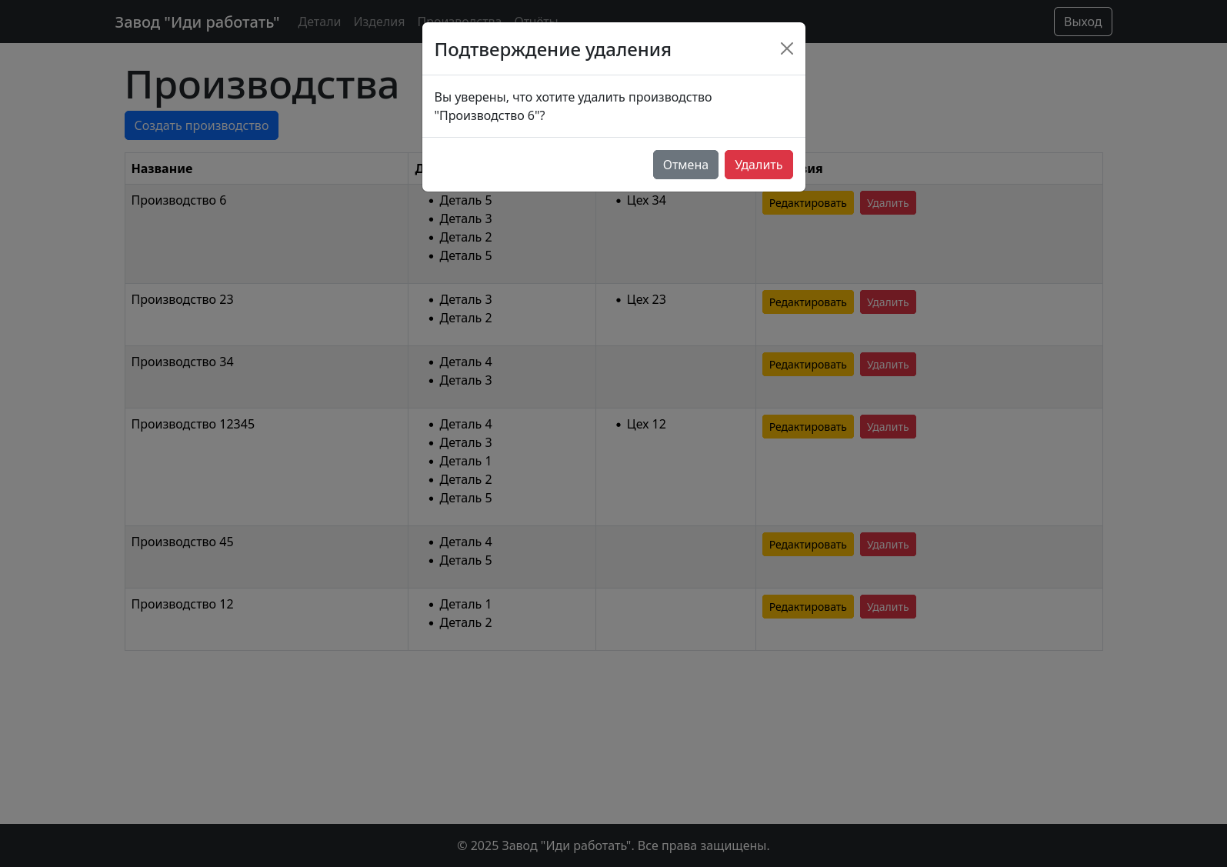
* 1. После нажатия на кнопку «Создать производство» откроется страница создания производства.



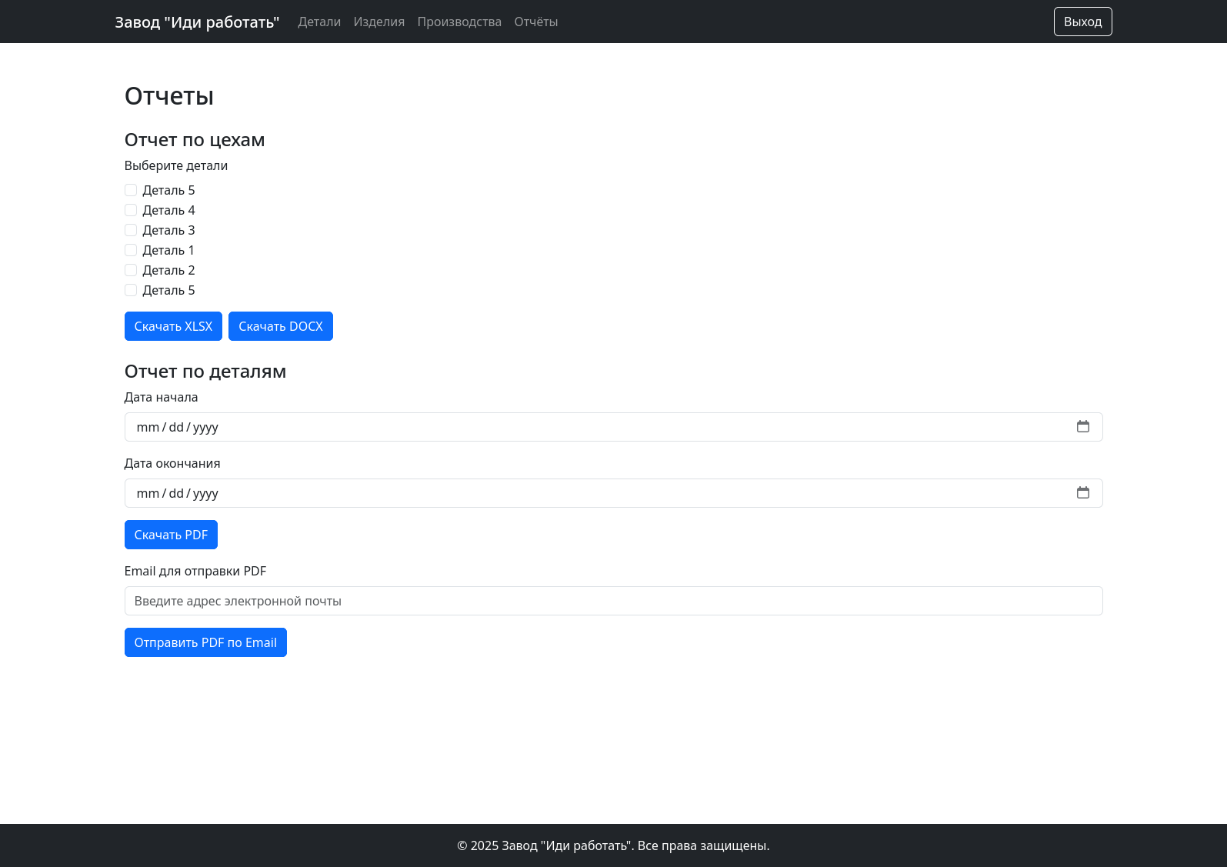
* 1. После нажатия на кнопку «Редактировать» откроется страница редактирования производства.



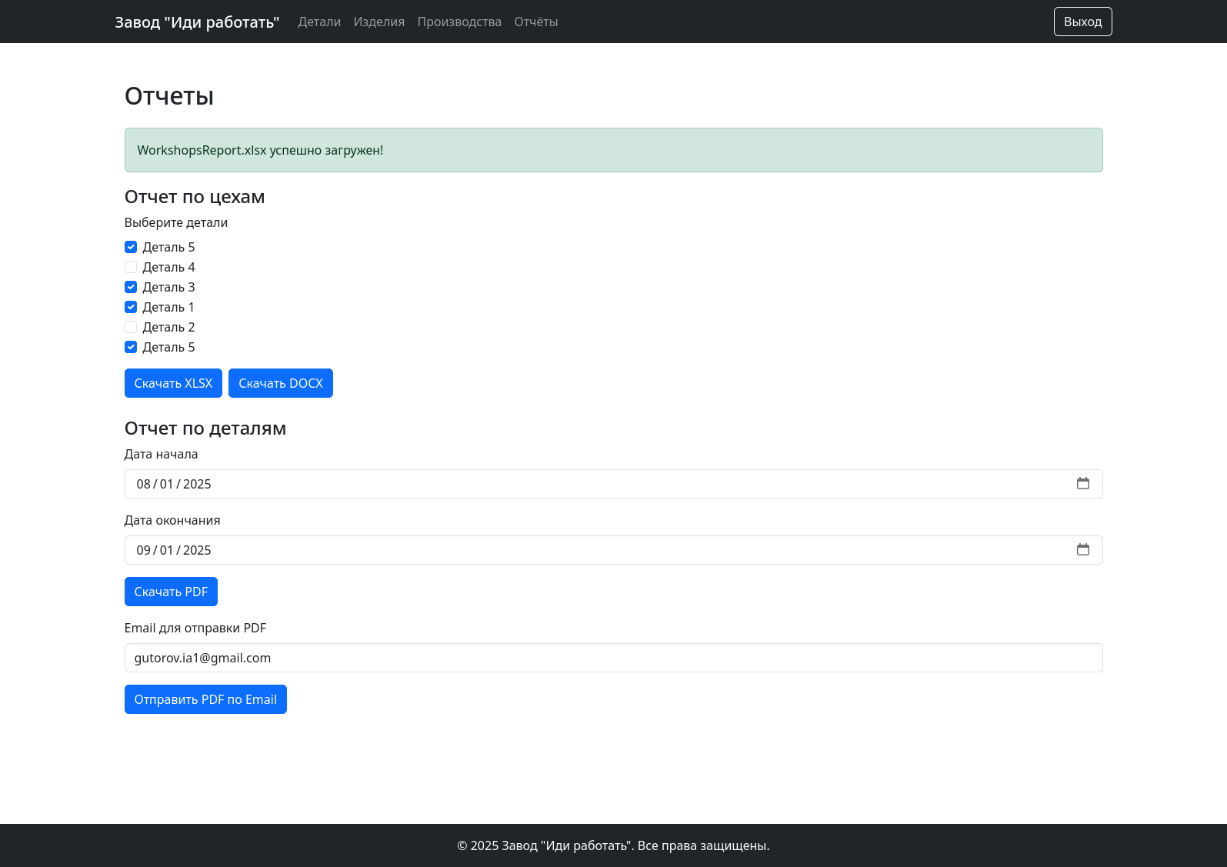
* 1. После нажатия на кнопку «Удалить» на производстве всплывёт окно подтверждения удаления производства.

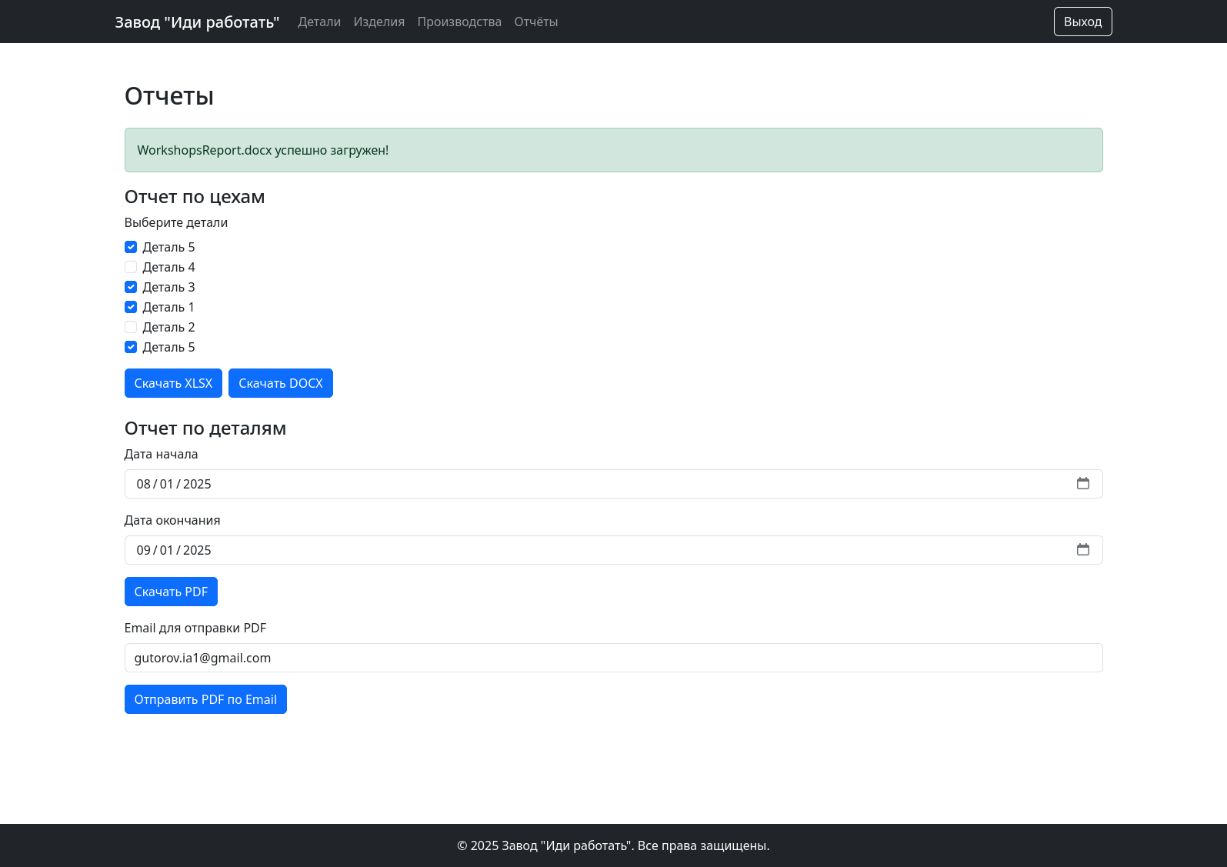


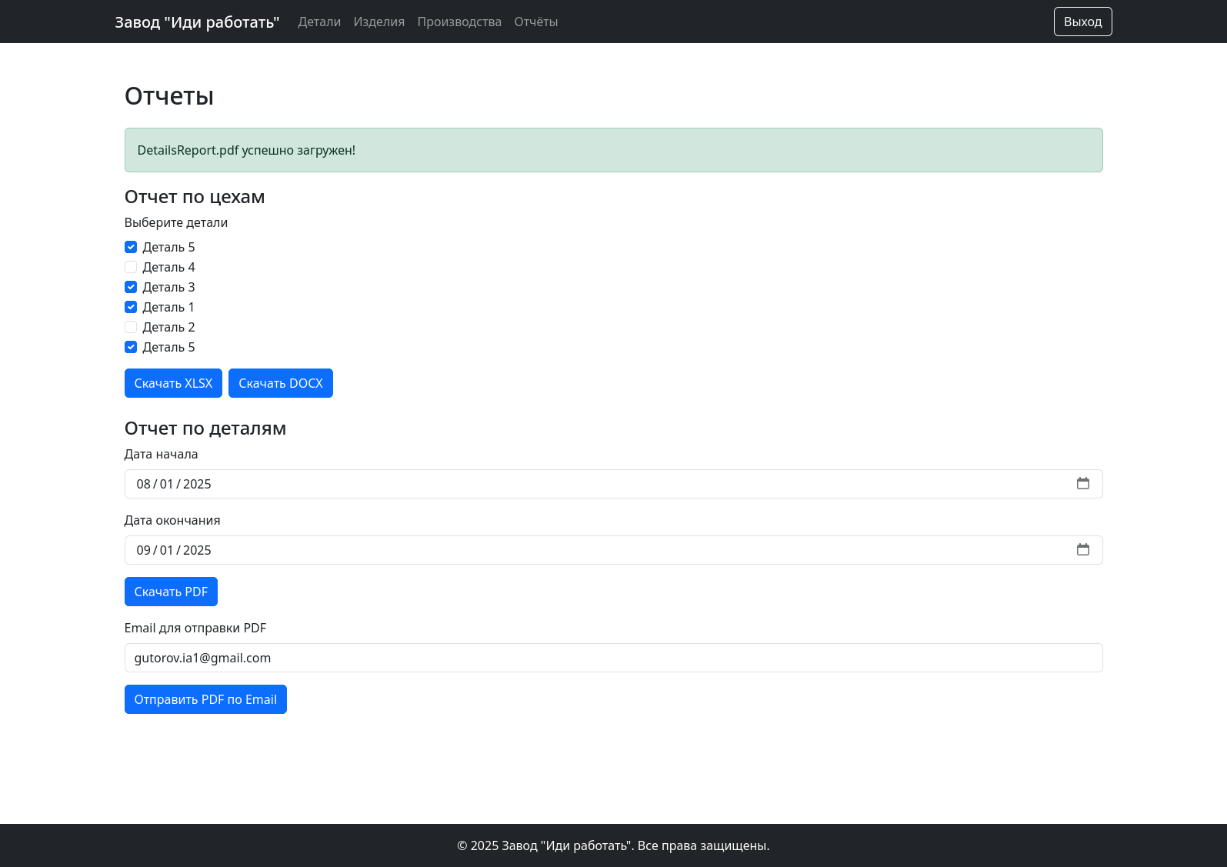
1. После нажатия на кнопку «Отчёты» откроется страница с отчётами.

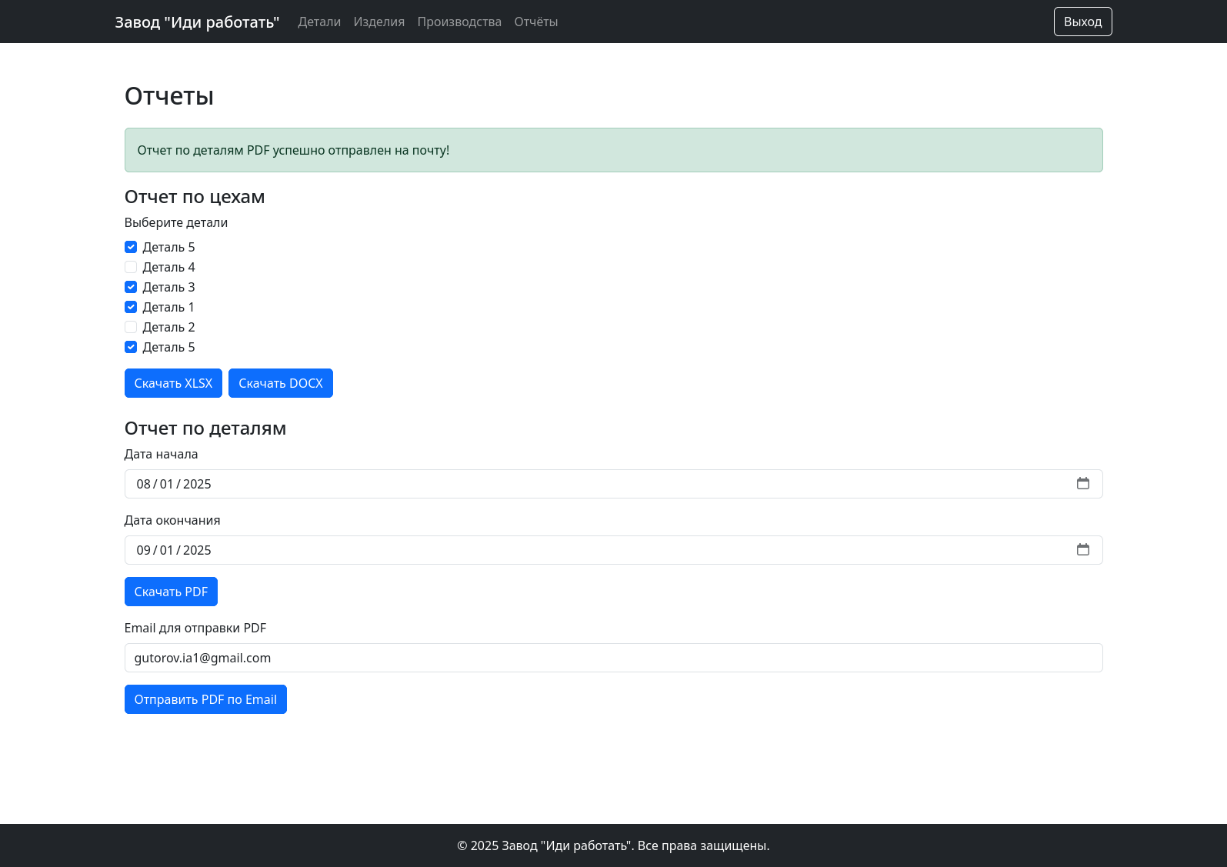


* 1. После заполнения и нажатия на кнопки «Скачать XLSX», «Скачать DOCX», «Скачать PDF» и «Отправить PDF по Email» появятся сообщения об успехе/неуспехе.

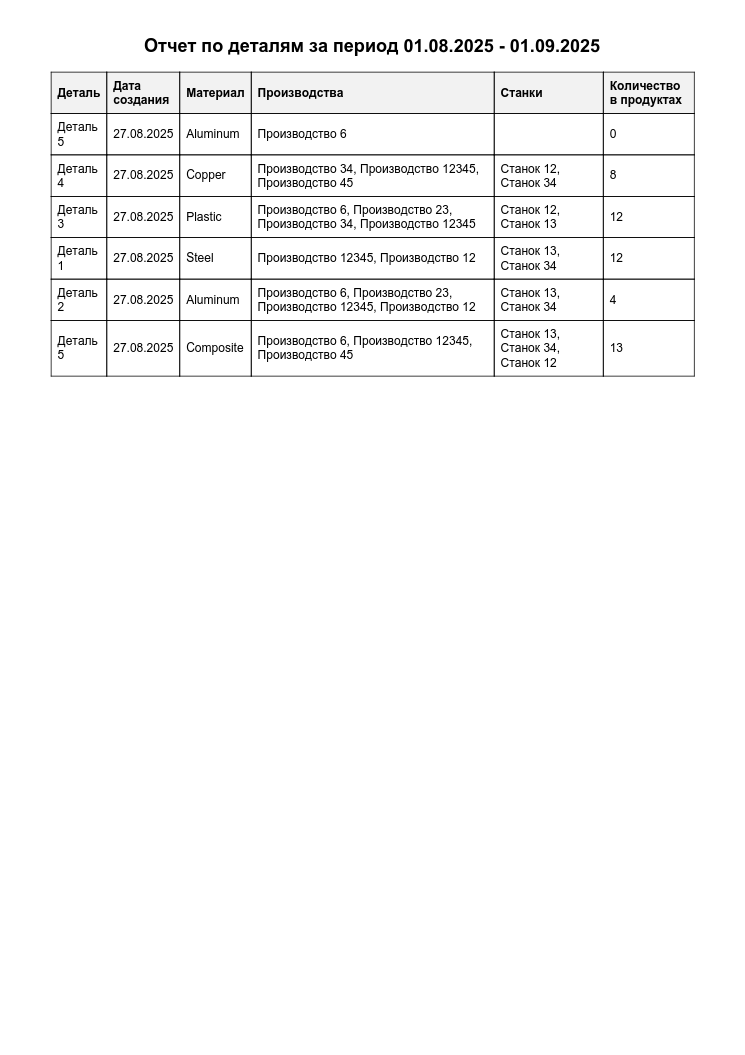


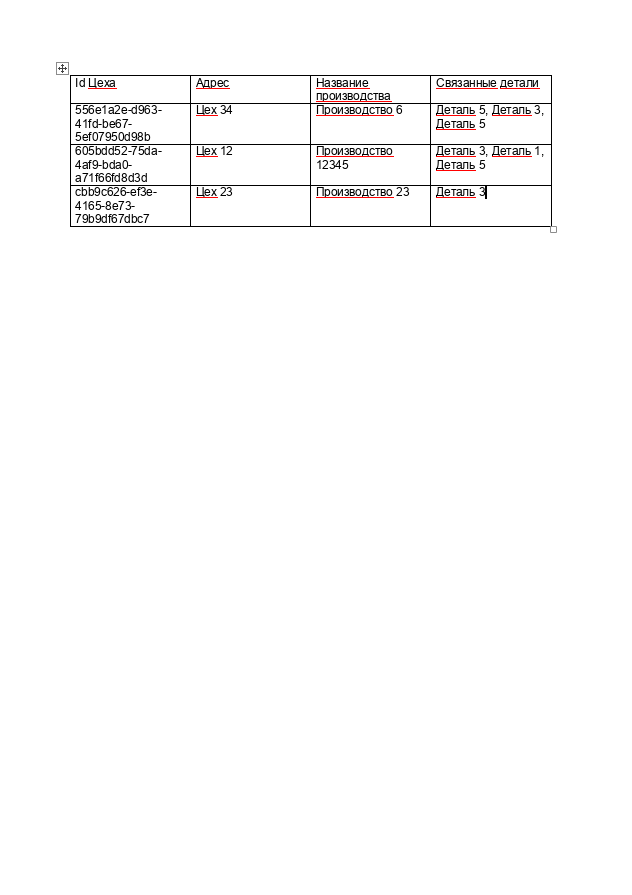


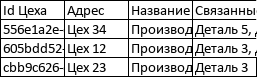


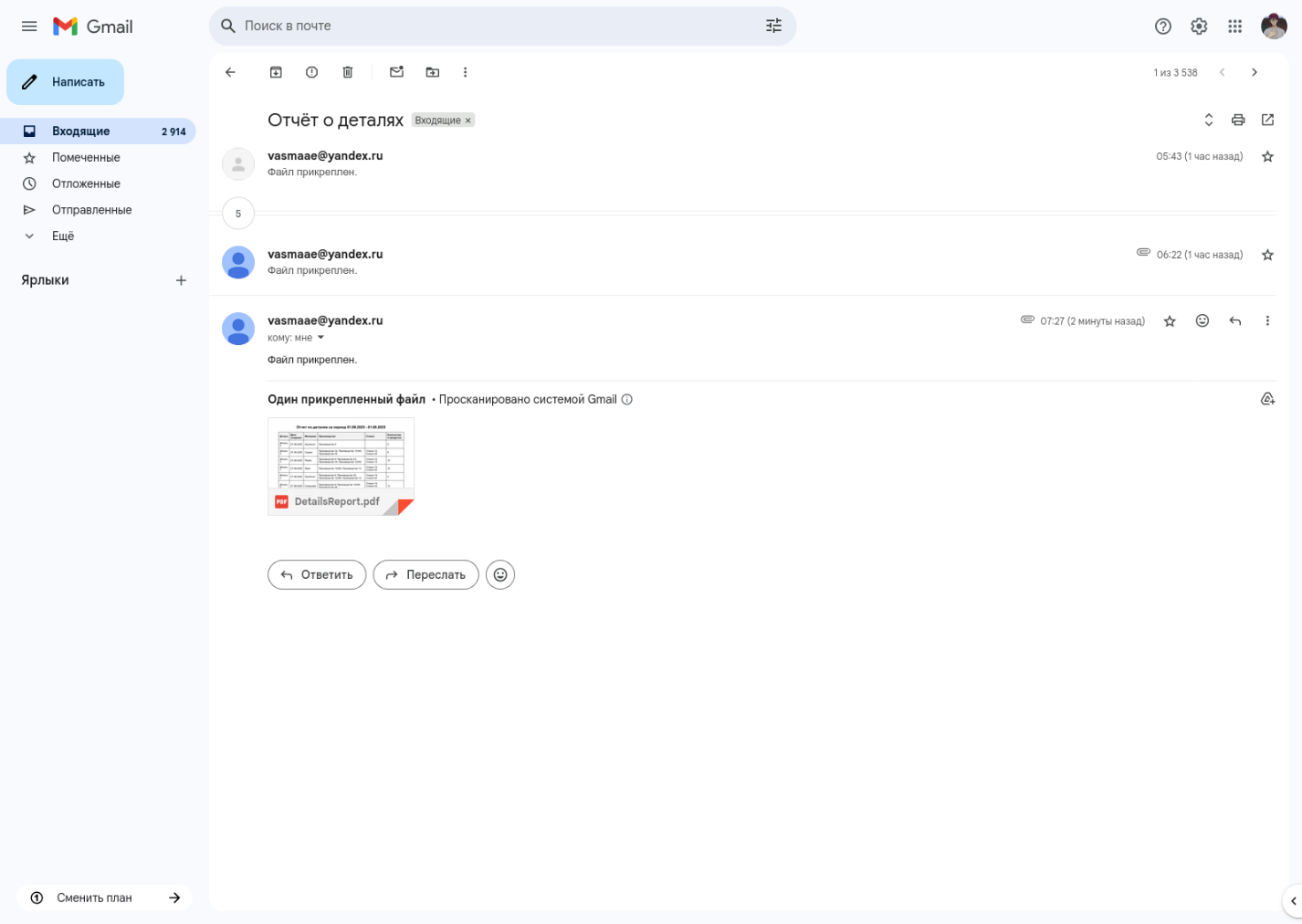


* 1. После скачивания отчётов они появятся на компьютере.









**Третья глава**

Руководство программиста на программный продукт «Завод «Иди работать». Исполнитель»

1. В проекте GoToWorkApi находится API-приложение на ASP.NET Core, реализующее контроллеры, адаптеры и инфраструктуру для взаимодействия с бизнес-логикой и базой данных.

1.1 В папке Adapters находятся адаптеры для работы с сущностями:

1.1.1 DetailAdapter.cs – адаптер деталей.  
1.1.2 EmployeeAdapter.cs – адаптер сотрудников.  
1.1.3 MachineAdapter.cs – адаптер станков.  
1.1.4 ProductAdapter.cs – адаптер продукции.  
1.1.5 ProductionAdapter.cs – адаптер производств.  
1.1.6 ReportAdapter.cs – адаптер отчетов.  
1.1.7 UserAdapter.cs – адаптер пользователей.  
1.1.8 WorkshopAdapter.cs – адаптер цехов.

1.2 В папке Controllers находятся контроллеры для REST API:

1.2.1 AuthController.cs – контроллер авторизации.  
1.2.2 DetailsController.cs – контроллер деталей.  
1.2.3 EmployeesController.cs – контроллер сотрудников.  
1.2.4 MachinesController.cs – контроллер станков.  
1.2.5 ProductionsController.cs – контроллер производств.  
1.2.6 ProductsController.cs – контроллер продукции.  
1.2.7 ReportsController.cs – контроллер отчетов.  
1.2.8 UsersController.cs – контроллер пользователей.  
1.2.9 WorkshopsController.cs – контроллер цехов.

1.3 В папке Infrastructure:

1.3.1 ConfigurationDatabase.cs – конфигурация базы данных.  
1.3.2 DatabaseSettings.cs – настройки подключения к базе.

1.4 Другие ключевые файлы:

* Program.cs – точка входа в приложение.
* AuthOptions.cs – параметры аутентификации и JWT-токенов.
* appsettings.json / appsettings.Development.json – конфигурация приложения.
* Dockerfile – описание контейнеризации API.

2. В проекте GoToWorkBusinessLogic находится бизнес-логика приложения.

2.1 В папке Implementations реализованы бизнес-логики:

2.1.1 DetailBusinessLogicContract.cs – логика деталей.  
2.1.2 EmployeeBusinessLogicContract.cs – логика сотрудников.  
2.1.3 MachineBusinessLogicContract.cs – логика станков.  
2.1.4 ProductBusinessLogicContract.cs – логика продукции.  
2.1.5 ProductionBusinessLogicContract.cs – логика производств.  
2.1.6 ReportContract.cs – логика отчетов.  
2.1.7 UserBusinessLogicContract.cs – логика пользователей.  
2.1.8 WorkshopBusinessLogicContract.cs – логика цехов.

3. В проекте GoToWorkContracts находятся контракты (интерфейсы, модели и вспомогательные классы).

3.1 В папке AdapterContracts:

3.1.1 IDetailAdapter.cs – интерфейс адаптера деталей.  
3.1.2 IEmployeeAdapter.cs – интерфейс адаптера сотрудников.  
3.1.3 IMachineAdapter.cs – интерфейс адаптера станков.  
3.1.4 IProductAdapter.cs – интерфейс адаптера продукции.  
3.1.5 IProductionAdapter.cs – интерфейс адаптера производств.  
3.1.6 IReportAdapter.cs – интерфейс адаптера отчетов.  
3.1.7 IUserAdapter.cs – интерфейс адаптера пользователей.  
3.1.8 IWorkshopAdapter.cs – интерфейс адаптера цехов.

3.2 В папке BindingModels – модели данных для передачи в API:

(детали, сотрудники, станки, продукция, пользователи, отчеты и др.).

3.3 В папке BusinessLogicContracts – интерфейсы бизнес-логики:

(для деталей, сотрудников, станков, продукции, производств, отчетов, пользователей, цехов и контракт для email).

3.4 В папке DataModels – модели данных для бизнес-логики:

(DetailDataModel, EmployeeDataModel, MachineDataModel, ProductDataModel, ProductionDataModel, UserDataModel, WorkshopDataModel и др.).

3.5 В папке Enums:

* MachineType.cs – типы станков.
* MaterialType.cs – типы материалов.
* UserRole.cs – роли пользователей.

3.6 В папке Exceptions – кастомные исключения:

(например, ElementNotFoundException, ValidationException, StorageException и др.).

3.7 В папке Extensions:

* RegexExtensions.cs – расширения для работы с регулярками.
* StringExtensions.cs – расширения для строк.

3.8 В папке Infrastructure:

* IConfigurationDatabase.cs – интерфейс конфигурации БД.
* IValidation.cs – интерфейс валидации.
* OperationResponse.cs – модель ответа операций.

3.9 В папке StoragesContracts – интерфейсы хранилищ данных:

(для деталей, сотрудников, станков, продукции, производств, пользователей, цехов).

3.10 В папке ViewModels – модели представления:

(для деталей, сотрудников, станков, продукции, производств, пользователей, отчетов, токенов, цехов и др.).

4. В проекте GoToWorkDatabase находится работа с базой данных (Entity Framework Core).

4.1 Основные файлы:

* GoToWorkDbContext.cs – контекст базы данных.
* DefaultConfigurationDatabase.cs – конфигурация базы по умолчанию.
* SampleContextFactory.cs – фабрика контекста.

4.2 В папке Implementations – реализации хранилищ:

(для деталей, сотрудников, станков, продукции, производств, пользователей, цехов).

4.3 В папке Migrations – миграции EF Core.

4.4 В папке Models – модели базы данных:

(Detail, Employee, Machine, Product, Production, User, Workshop и связи между ними).

5. В проекте GoToWorkWeb находится фронтенд (React + TypeScript + Vite).

5.1 В папке components – компоненты интерфейса:

(формы и страницы для деталей, сотрудников, станков, продукции, производств, отчетов, пользователей, цехов, а также HomePage, LoginPage, RegisterPage).

5.2 В папке layouts:

* MainLayout.tsx – основной шаблон приложения.
* Footer.tsx – подвал сайта.

5.3 В папке router:

* AppRouter.tsx – маршрутизация приложения.

5.4 В папке services – сервисы для работы с API:

(authService, detailService, employeeService, machineService, productionService, productService, reportService, workshopService).

5.5 В папке types:

* types.ts – типы данных для TypeScript.

5.6 Основные файлы:

* App.tsx – главный компонент приложения.
* main.tsx – точка входа.
* index.css – глобальные стили.
* vite-env.d.ts – настройки окружения Vite.

**Заключение**

Курсовая работа на тему «Разработка программы “Завод “Иди работать”. Исполнитель”» успешно завершена: в результате проведённого исследования был разработан программный продукт, который позволяет автоматизировать ключевые процессы управления производственными заданиями — от их постановки и назначения исполнителям до контроля выполнения и формирования отчётности. Внедрение данной системы на предприятии позволит существенно повысить эффективность труда, сократить издержки, связанные с ручным планированием, и обеспечить прозрачность управления задачами, что подтверждает практическую значимость проекта и достижение поставленной цели.

**Список литературы**

1. ProfessorWeb. .Net & Web Programming [Электронный ресурс]/ Режим доступа: https://professorweb.ru/ . – Загл. с экрана.
2. METANIT.COM. Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com/sharp/ . – Загл. с экрана.
3. Microsoft Learn. Официальный сайт Microsoft [Электронный ресурс] / Режим доступа: https://learn.microsoft.com/. – Загл. с экрана.
4. StackOverflow. Форум для программистов [Электронный ресурс] / Режим доступа: <https://ru.stackoverflow.com/>. – Загл. с экрана.
5. Бабич А.В. Введение в UML [Электронный ресурс]/ Бабич А.В. Режим доступа: https://www.intuit.ru/goods\_store/ebooks/8359.
6. Informicus. Диаграммы вариантов использования [Электронный ресурс]/ Режим доступа: http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=4 . – Загл. с экрана
7. Теория и практика UML. Диаграмма последовательности [Электронный ресурс] / Режим доступа: http://it-gost.ru/articles/view\_articles/94 . – Загл. с экрана.
8. Pro Git. 2nd Edition [Электронный ресурс] / Режим доступа: https://git-scm.com/book/ru/v2
9. Nicepage. Все о работе с CSS и HTML [Электронный ресурс] / Режим доступа: https://nicepage.com/
10. Хабр Q&A — вопросы и ответы для IT-специалистов [Электронный ресурс] / Режим доступа: https://qna.habr.com/

Приложение. Листинг кода.

|  |
| --- |
| // SampleContextFactory.cs  using Microsoft.EntityFrameworkCore.Design;  namespace GoToWorkDatabase;  internal class SampleContextFactory : IDesignTimeDbContextFactory<GoToWorkDbContext>  {  public GoToWorkDbContext CreateDbContext(string[] args)  {  return new GoToWorkDbContext(new DefaultConfigurationDatabase());  }  }  // GoToWorkDbContext.cs  using GoToWorkContracts.Infrastructure;  using GoToWorkDatabase.Models;  using Microsoft.EntityFrameworkCore;  namespace GoToWorkDatabase;  public class GoToWorkDbContext : DbContext  {  private readonly IConfigurationDatabase? \_configurationDatabase;  public GoToWorkDbContext(IConfigurationDatabase? configurationDatabase)  {  \_configurationDatabase = configurationDatabase;  }  public DbSet<Detail> Details { get; set; }  public DbSet<DetailProduct> DetailProducts { get; set; }  public DbSet<DetailProduction> DetailProductions { get; set; }  public DbSet<Employee> Employees { get; set; }  public DbSet<EmployeeMachine> EmployeeMachines { get; set; }  public DbSet<EmployeeWorkshop> EmployeeWorkshops { get; set; }  public DbSet<Machine> Machines { get; set; }  public DbSet<Production> Productions { get; set; }  // SampleContextFactory.cs  using Microsoft.EntityFrameworkCore.Design;  namespace GoToWorkDatabase;  internal class SampleContextFactory : IDesignTimeDbContextFactory<GoToWorkDbContext>  {  public GoToWorkDbContext CreateDbContext(string[] args)  {  return new GoToWorkDbContext(new DefaultConfigurationDatabase());  }  }  // GoToWorkDbContext.cs  using GoToWorkContracts.Infrastructure;  using GoToWorkDatabase.Models;  using Microsoft.EntityFrameworkCore;  namespace GoToWorkDatabase;  public class GoToWorkDbContext : DbContext  {  private readonly IConfigurationDatabase? \_configurationDatabase;  public GoToWorkDbContext(IConfigurationDatabase? configurationDatabase)  {  \_configurationDatabase = configurationDatabase;  }  public DbSet<Detail> Details { get; set; }  public DbSet<DetailProduct> DetailProducts { get; set; }  public DbSet<DetailProduction> DetailProductions { get; set; }  public DbSet<Employee> Employees { get; set; }  public DbSet<EmployeeMachine> EmployeeMachines { get; set; }  public DbSet<EmployeeWorkshop> EmployeeWorkshops { get; set; }  public DbSet<Machine> Machines { get; set; }  public DbSet<Production> Productions { get; set; }  // SampleContextFactory.cs  using Microsoft.EntityFrameworkCore.Design;  namespace GoToWorkDatabase;  internal class SampleContextFactory : IDesignTimeDbContextFactory<GoToWorkDbContext>  {  public GoToWorkDbContext CreateDbContext(string[] args)  {  return new GoToWorkDbContext(new DefaultConfigurationDatabase());  }  }  // GoToWorkDbContext.cs  using GoToWorkContracts.Infrastructure;  using GoToWorkDatabase.Models;  using Microsoft.EntityFrameworkCore;  namespace GoToWorkDatabase;  public class GoToWorkDbContext : DbContext  {  private readonly IConfigurationDatabase? \_configurationDatabase;  public GoToWorkDbContext(IConfigurationDatabase? configurationDatabase)  {  \_configurationDatabase = configurationDatabase;  }  public DbSet<Detail> Details { get; set; }  public DbSet<DetailProduct> DetailProducts { get; set; }  public DbSet<DetailProduction> DetailProductions { get; set; }  public DbSet<Employee> Employees { get; set; }  public DbSet<EmployeeMachine> EmployeeMachines { get; set; }  public DbSet<EmployeeWorkshop> EmployeeWorkshops { get; set; }  public DbSet<Machine> Machines { get; set; }  public DbSet<Production> Productions { get; set; }  // SampleContextFactory.cs  using Microsoft.EntityFrameworkCore.Design;  namespace GoToWorkDatabase;  internal class SampleContextFactory : IDesignTimeDbContextFactory<GoToWorkDbContext>  {  public GoToWorkDbContext CreateDbContext(string[] args)  {  return new GoToWorkDbContext(new DefaultConfigurationDatabase());  }  }  // GoToWorkDbContext.cs  using GoToWorkContracts.Infrastructure;  using GoToWorkDatabase.Models;  using Microsoft.EntityFrameworkCore;  namespace GoToWorkDatabase;  public class GoToWorkDbContext : DbContext  {  private readonly IConfigurationDatabase? \_configurationDatabase;  public GoToWorkDbContext(IConfigurationDatabase? configurationDatabase)  {  \_configurationDatabase = configurationDatabase;  }  public DbSet<Detail> Details { get; set; }  public DbSet<DetailProduct> DetailProducts { get; set; }  public DbSet<DetailProduction> DetailProductions { get; set; }  public DbSet<Employee> Employees { get; set; }  public DbSet<EmployeeMachine> EmployeeMachines { get; set; }  public DbSet<EmployeeWorkshop> EmployeeWorkshops { get; set; }  public DbSet<Machine> Machines { get; set; }  public DbSet<Production> Productions { get; set; }  // SampleContextFactory.cs  using Microsoft.EntityFrameworkCore.Design;  namespace GoToWorkDatabase;  internal class SampleContextFactory : IDesignTimeDbContextFactory<GoToWorkDbContext>  {  public GoToWorkDbContext CreateDbContext(string[] args)  {  return new GoToWorkDbContext(new DefaultConfigurationDatabase());  }  }  // GoToWorkDbContext.cs  using GoToWorkContracts.Infrastructure;  using GoToWorkDatabase.Models;  using Microsoft.EntityFrameworkCore;  namespace GoToWorkDatabase;  public class GoToWorkDbContext : DbContext  {  private readonly IConfigurationDatabase? \_configurationDatabase;  public GoToWorkDbContext(IConfigurationDatabase? configurationDatabase)  {  \_configurationDatabase = configurationDatabase;  }  public DbSet<Detail> Details { get; set; }  public DbSet<DetailProduct> DetailProducts { get; set; }  public DbSet<DetailProduction> DetailProductions { get; set; }  public DbSet<Employee> Employees { get; set; }  public DbSet<EmployeeMachine> EmployeeMachines { get; set; }  public DbSet<EmployeeWorkshop> EmployeeWorkshops { get; set; }  public DbSet<Machine> Machines { get; set; }  public DbSet<Production> Productions { get; set; }  public DbSet<Product> Products { get; set; }  public DbSet<Workshop> Workshops { get; set; }  public DbSet<User> Users { get; set; }  protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  {  optionsBuilder.UseNpgsql(\_configurationDatabase?.ConnectionString, o => o.SetPostgresVersion(12, 2));  base.OnConfiguring(optionsBuilder);  }  protected override void OnModelCreating(ModelBuilder modelBuilder)  {  base.OnModelCreating(modelBuilder);  modelBuilder.Entity<User>()  .HasIndex(u => u.Login)  .IsUnique();  modelBuilder.Entity<User>()  .HasIndex(u => u.Email)  .IsUnique();  modelBuilder.Entity<Workshop>()  .HasIndex(w => w.Address)  .IsUnique();  modelBuilder.Entity<Product>()  .HasIndex(p => p.Name)  .IsUnique();  modelBuilder.Entity<Production>()  .HasIndex(p => p.Name)  .IsUnique();  modelBuilder.Entity<Machine>()  .HasIndex(m => m.Model)  .IsUnique();  modelBuilder.Entity<Machine>()  .HasMany(m => m.Products)  .WithOne(p => p.Machine)  .HasForeignKey(m => m.MachineId);  modelBuilder.Entity<Production>()  .HasMany(p => p.Workshops)  .WithOne(w => w.Production)  .HasForeignKey(w => w.ProductionId);  modelBuilder.Entity<DetailProduct>()  .HasKey(dp => new { dp.DetailId, dp.ProductId });  modelBuilder.Entity<DetailProduction>()  .HasKey(dp => new { dp.DetailId, dp.ProductionId });  modelBuilder.Entity<EmployeeMachine>()  .HasKey(em => new { em.EmployeeId, em.MachineId });  modelBuilder.Entity<EmployeeWorkshop>()  .HasKey(ew => new { ew.EmployeeId, ew.WorkshopId });  }  }  // DefaultConfigurationDatabase.cs  using GoToWorkContracts.Infrastructure;  namespace GoToWorkDatabase;  internal class DefaultConfigurationDatabase : IConfigurationDatabase  {  public string ConnectionString => "Host=127.0.0.1;Port=5432;Database=GoToWork;Username=postgres;Password=postgres;";  }  // DetailStorageContract.cs  using AutoMapper;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.StoragesContracts;  using GoToWorkDatabase.Models;  using Microsoft.EntityFrameworkCore;  namespace GoToWorkDatabase.Implementations;  internal class DetailStorageContract : IDetailStorageContract  {  private readonly GoToWorkDbContext \_dbContext;  private readonly Mapper \_mapper;  public DetailStorageContract(GoToWorkDbContext dbContext)  {  \_dbContext = dbContext;  var config = new MapperConfiguration(cfg =>  {  cfg.CreateMap<Detail, DetailDataModel>();  cfg.CreateMap<DetailDataModel, Detail>();  cfg.CreateMap<Product, ProductDataModel>();  cfg.CreateMap<ProductDataModel, Product>();  cfg.CreateMap<Production, ProductionDataModel>();  cfg.CreateMap<ProductionDataModel, Production>();  cfg.CreateMap<DetailProduct, DetailProductDataModel>();  cfg.CreateMap<DetailProductDataModel, DetailProduct>();  cfg.CreateMap<DetailProduction, DetailProductionDataModel>();  cfg.CreateMap<DetailProductionDataModel, DetailProduction>();  });  \_mapper = new Mapper(config);  }  public List<DetailDataModel> GetList(DateTime? startDate = null, DateTime? endDate = null)  {  try  {  var query = \_dbContext.Details  .Include(x => x.DetailProducts)!  .ThenInclude(x => x.Product)  .Include(x => x.DetailProductions)!  .ThenInclude(x => x.Production)  .AsQueryable();  if (startDate is not null)  query = query.Where(x => x.CreationDate >= startDate);  if (endDate is not null)  query = query.Where(x => x.CreationDate <= endDate);  return [.. query.Select(d => \_mapper.Map<DetailDataModel>(d))];  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public DetailDataModel? GetElementById(string id)  {  try  {  return \_mapper.Map<DetailDataModel>(GetDetailById(id));  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public DetailDataModel? GetElementByName(string name)  {  try  {  return \_mapper.Map<DetailDataModel>(\_dbContext.Details  .Include(x => x.DetailProducts)!  .ThenInclude(x => x.Product)  .Include(x => x.DetailProductions)!  .ThenInclude(x => x.Production).FirstOrDefault(d => d.Name == name));  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void AddElement(DetailDataModel detailDataModel)  {  try  {  \_dbContext.Details.Add(\_mapper.Map<Detail>(detailDataModel));  \_dbContext.SaveChanges();  }  catch (InvalidOperationException ex) when (ex.TargetSite?.Name == "ThrowIdentityConflict")  {  \_dbContext.ChangeTracker.Clear();  throw new ElementExistsException("Id", detailDataModel.Id);  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void UpdElement(DetailDataModel detailDataModel)  {  try  {  var element = GetDetailById(detailDataModel.Id)  ?? throw new ElementNotFoundException(detailDataModel.Id);  \_dbContext.Details.Update(\_mapper.Map(detailDataModel, element));  \_dbContext.SaveChanges();  }  catch (ElementNotFoundException ex)  {  \_dbContext.ChangeTracker.Clear();  throw;  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void DelElement(string id)  {  try  {  var element = GetDetailById(id) ?? throw new ElementNotFoundException(id);  \_dbContext.Details.Remove(element);  \_dbContext.SaveChanges();  }  catch (Exception ex) when (ex is ElementNotFoundException)  {  \_dbContext.ChangeTracker.Clear();  throw;  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  private Detail? GetDetailById(string id) => \_dbContext.Details  .Include(x => x.DetailProducts)!  .ThenInclude(x => x.Product)  .Include(x => x.DetailProductions)!  .ThenInclude(x => x.Production)  .FirstOrDefault(d => d.Id == id);  }  // WorkshopStorageContract.cs  using AutoMapper;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.StoragesContracts;  using GoToWorkDatabase.Models;  using Microsoft.EntityFrameworkCore;  namespace GoToWorkDatabase.Implementations;  internal class WorkshopStorageContract : IWorkshopStorageContract  {  private readonly GoToWorkDbContext \_dbContext;  private readonly Mapper \_mapper;  public WorkshopStorageContract(GoToWorkDbContext dbContext)  {  \_dbContext = dbContext;  var config = new MapperConfiguration(cfg =>  {  cfg.CreateMap<Workshop, WorkshopDataModel>()  .ForMember(dest => dest.Employees, opt => opt.MapFrom(src => src.EmployeeWorkshops));  cfg.CreateMap<WorkshopDataModel, Workshop>()  .ForMember(dest => dest.EmployeeWorkshops, opt => opt.MapFrom(src => src.Employees));  cfg.CreateMap<Production, ProductionDataModel>();  cfg.CreateMap<ProductionDataModel, Production>();  cfg.CreateMap<Employee, EmployeeDataModel>();  cfg.CreateMap<EmployeeDataModel, Employee>();  cfg.CreateMap<EmployeeWorkshop, EmployeeWorkshopDataModel>();  cfg.CreateMap<EmployeeWorkshopDataModel, EmployeeWorkshop>();  });  \_mapper = new Mapper(config);  }  public List<WorkshopDataModel> GetList(string? productionId = null)  {  try  {  var query = \_dbContext.Workshops  .Include(x => x.Production)  .Include(x => x.EmployeeWorkshops)!  .ThenInclude(x => x.Employee)  .AsQueryable();  if (productionId is not null)  query = query.Where(w => w.ProductionId == productionId);  return [.. query.Select(w => \_mapper.Map<WorkshopDataModel>(w))];  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public WorkshopDataModel? GetElementById(string id)  {  try  {  return \_mapper.Map<WorkshopDataModel>(GetWorkshopById(id));  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public WorkshopDataModel? GetElementByAddress(string address)  {  try  {  var workshop = \_dbContext.Workshops  .Include(x => x.Production)  .Include(x => x.EmployeeWorkshops)!  .ThenInclude(x => x.Employee)  .FirstOrDefault(w => w.Address == address);  return \_mapper.Map<WorkshopDataModel>(workshop);  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void AddElement(WorkshopDataModel workshopDataModel)  {  try  {  \_dbContext.Workshops.Add(\_mapper.Map<Workshop>(workshopDataModel));  \_dbContext.SaveChanges();  }  catch (InvalidOperationException ex) when (ex.TargetSite?.Name == "ThrowIdentityConflict")  {  \_dbContext.ChangeTracker.Clear();  throw new ElementExistsException("Id", workshopDataModel.Id);  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void UpdElement(WorkshopDataModel workshopDataModel)  {  try  {  var existingWorkshop = GetWorkshopById(workshopDataModel.Id) ??  throw new ElementNotFoundException(workshopDataModel.Id);  \_dbContext.Workshops.Update(\_mapper.Map(workshopDataModel, existingWorkshop));  \_dbContext.SaveChanges();  }  catch (Exception ex) when (ex is ElementNotFoundException)  {  \_dbContext.ChangeTracker.Clear();  throw;  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void DelElement(string id)  {  try  {  var workshop = GetWorkshopById(id) ?? throw new ElementNotFoundException(id);  \_dbContext.Workshops.Remove(workshop);  \_dbContext.SaveChanges();  }  catch (Exception ex) when (ex is ElementNotFoundException)  {  \_dbContext.ChangeTracker.Clear();  throw;  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  private Workshop? GetWorkshopById(string id) => \_dbContext.Workshops  .Include(x => x.Production)  .Include(x => x.EmployeeWorkshops)!  .ThenInclude(x => x.Employee)  .FirstOrDefault(w => w.Id == id);  }  // EmployeeStorageContract.cs  using AutoMapper;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.StoragesContracts;  using GoToWorkContracts.ViewModels;  using GoToWorkDatabase.Models;  using Microsoft.EntityFrameworkCore;  namespace GoToWorkDatabase.Implementations;  internal class EmployeeStorageContract : IEmployeeStorageContract  {  private readonly GoToWorkDbContext \_dbContext;  private readonly Mapper \_mapper;  public EmployeeStorageContract(GoToWorkDbContext dbContext)  {  \_dbContext = dbContext;  var config = new MapperConfiguration(cfg =>  {  cfg.CreateMap<Employee, EmployeeDataModel>();  cfg.CreateMap<EmployeeDataModel, Employee>();  cfg.CreateMap<Machine, MachineDataModel>();  cfg.CreateMap<MachineDataModel, Machine>();  cfg.CreateMap<Workshop, WorkshopDataModel>();  cfg.CreateMap<WorkshopDataModel, Workshop>();  cfg.CreateMap<EmployeeMachine, EmployeeMachineDataModel>();  cfg.CreateMap<EmployeeMachineDataModel, EmployeeMachine>();  cfg.CreateMap<EmployeeWorkshop, EmployeeWorkshopDataModel>();  cfg.CreateMap<EmployeeWorkshopDataModel, EmployeeWorkshop>();  });  \_mapper = new Mapper(config);  }  public List<EmployeeDataModel> GetList()  {  try  {  return  [  .. \_dbContext.Employees  .Include(x => x.EmployeeMachines)!  .ThenInclude(x => x.Machine)  .Include(x => x.EmployeeWorkshops)!  .ThenInclude(x => x.Workshop)  .Select(x => \_mapper.Map<EmployeeDataModel>(x))  ];  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public EmployeeDataModel? GetElementById(string id)  {  try  {  return \_mapper.Map<EmployeeDataModel>(GetEmployeeById(id));  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public List<EmployeeDataModel> GetElementsByFullName(string fullName)  {  try  {  return  [  .. \_dbContext.Employees  .Include(x => x.EmployeeMachines)!  .ThenInclude(x => x.Machine)  .Include(x => x.EmployeeWorkshops)!  .ThenInclude(x => x.Workshop)  .Where(e => e.FullName == fullName)  .Select(x => \_mapper.Map<EmployeeDataModel>(x))  ];  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void AddElement(EmployeeDataModel employeeDataModel)  {  try  {  \_dbContext.Employees.Add(\_mapper.Map<Employee>(employeeDataModel));  \_dbContext.SaveChanges();  }  catch (InvalidOperationException ex) when (ex.TargetSite?.Name == "ThrowIdentityConflict")  {  \_dbContext.ChangeTracker.Clear();  throw new ElementExistsException("Id", employeeDataModel.Id);  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void UpdElement(EmployeeDataModel employeeDataModel)  {  try  {  var existing = GetEmployeeById(employeeDataModel.Id) ??  throw new ElementNotFoundException(employeeDataModel.Id);  \_dbContext.Employees.Update(\_mapper.Map(employeeDataModel, existing));  \_dbContext.SaveChanges();  }  catch (ElementNotFoundException ex)  {  \_dbContext.ChangeTracker.Clear();  throw;  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void DelElement(string id)  {  try  {  var element = GetEmployeeById(id) ?? throw new ElementNotFoundException(id);  \_dbContext.Employees.Remove(element);  \_dbContext.SaveChanges();  }  catch (ElementNotFoundException ex)  {  \_dbContext.ChangeTracker.Clear();  throw;  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  private Employee? GetEmployeeById(string id) => \_dbContext.Employees  .Include(x => x.EmployeeMachines)!  .ThenInclude(x => x.Machine)  .Include(x => x.EmployeeWorkshops)!  .ThenInclude(x => x.Workshop)  .FirstOrDefault(e => e.Id == id);  }  // ProductionStorageContract.cs  using AutoMapper;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.StoragesContracts;  using GoToWorkDatabase.Models;  using Microsoft.EntityFrameworkCore;  namespace GoToWorkDatabase.Implementations;  internal class ProductionStorageContract : IProductionStorageContract  {  private readonly GoToWorkDbContext \_dbContext;  private readonly Mapper \_mapper;  public ProductionStorageContract(GoToWorkDbContext dbContext)  {  \_dbContext = dbContext;  var config = new MapperConfiguration(cfg =>  {  cfg.CreateMap<Production, ProductionDataModel>()  .ForMember(dest => dest.Details, opt => opt.MapFrom(x => x.DetailProductions));  cfg.CreateMap<ProductionDataModel, Production>()  .ForMember(dest => dest.DetailProductions, opt => opt.MapFrom(x => x.Details));  cfg.CreateMap<Workshop, WorkshopDataModel>();  cfg.CreateMap<WorkshopDataModel, Workshop>();  cfg.CreateMap<Detail, DetailDataModel>();  cfg.CreateMap<DetailDataModel, Detail>();  cfg.CreateMap<DetailProduction, DetailProductionDataModel>();  cfg.CreateMap<DetailProductionDataModel, DetailProduction>();  });  \_mapper = new Mapper(config);  }  public List<ProductionDataModel> GetList()  {  try  {  return  [  ..\_dbContext.Productions  .Include(p => p.Workshops)  .Include(p => p.DetailProductions)!  .ThenInclude(d => d.Detail)  .Select(p => \_mapper.Map<ProductionDataModel>(p))  ];  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public ProductionDataModel? GetElementById(string id)  {  try  {  return \_mapper.Map<ProductionDataModel>(GetProductionById(id));  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public ProductionDataModel? GetElementByName(string name)  {  try  {  return \_mapper.Map<ProductionDataModel>(\_dbContext.Productions.FirstOrDefault(p => p.Name == name));  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void AddElement(ProductionDataModel productionDataModel)  {  try  {  \_dbContext.Productions.Add(\_mapper.Map<Production>(productionDataModel));  \_dbContext.SaveChanges();  }  catch (InvalidOperationException ex) when (ex.TargetSite?.Name == "ThrowIdentityConflict")  {  \_dbContext.ChangeTracker.Clear();  throw new ElementExistsException("Id", productionDataModel.Id);  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void UpdElement(ProductionDataModel productionDataModel)  {  try  {  var existingProduction = GetProductionById(productionDataModel.Id)  ?? throw new ElementNotFoundException(productionDataModel.Id);  \_dbContext.Productions.Update(\_mapper.Map(productionDataModel, existingProduction));  \_dbContext.SaveChanges();  }  catch (Exception ex) when (ex is ElementNotFoundException)  {  \_dbContext.ChangeTracker.Clear();  throw;  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void DelElement(string id)  {  try  {  var production = GetProductionById(id) ?? throw new ElementNotFoundException(id);  \_dbContext.Productions.Remove(production);  \_dbContext.SaveChanges();  }  catch (Exception ex) when (ex is ElementNotFoundException)  {  \_dbContext.ChangeTracker.Clear();  throw;  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  private Production? GetProductionById(string id) => \_dbContext.Productions  .Include(p => p.Workshops)  .Include(p => p.DetailProductions)!  .ThenInclude(d => d.Detail)  .FirstOrDefault(p => p.Id == id);  }  // ProductStorageContract.cs  using AutoMapper;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.StoragesContracts;  using GoToWorkDatabase.Models;  using Microsoft.EntityFrameworkCore;  namespace GoToWorkDatabase.Implementations;  internal class ProductStorageContract : IProductStorageContract  {  private readonly GoToWorkDbContext \_dbContext;  private readonly Mapper \_mapper;  public ProductStorageContract(GoToWorkDbContext dbContext)  {  \_dbContext = dbContext;  var config = new MapperConfiguration(cfg =>  {  cfg.CreateMap<Product, ProductDataModel>()  .ForMember(dest => dest.Details, opt => opt.MapFrom(x => x.DetailProducts));  cfg.CreateMap<ProductDataModel, Product>()  .ForMember(dest => dest.DetailProducts, opt => opt.MapFrom(x => x.Details));  cfg.CreateMap<Machine, MachineDataModel>();  cfg.CreateMap<MachineDataModel, Machine>();  cfg.CreateMap<Detail, DetailDataModel>();  cfg.CreateMap<DetailDataModel, Detail>();  cfg.CreateMap<DetailProduct, DetailProductDataModel>();  cfg.CreateMap<DetailProductDataModel, DetailProduct>();  });  \_mapper = new Mapper(config);  }  public List<ProductDataModel> GetList(DateTime? startDate = null, DateTime? endDate = null,  string? machineId = null)  {  try  {  var query = \_dbContext.Products  .Include(x => x.Machine)  .Include(x => x.DetailProducts)!  .ThenInclude(x => x.Detail)  .AsQueryable();  if (startDate is not null)  query = query.Where(p => p.CreationDate >= startDate.Value);  if (endDate is not null)  query = query.Where(p => p.CreationDate <= endDate.Value);  if (machineId is not null)  query = query.Where(p => p.MachineId == machineId);  return [..query.Select(p => \_mapper.Map<ProductDataModel>(p))];  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public ProductDataModel? GetElementById(string id)  {  try  {  return \_mapper.Map<ProductDataModel>(GetProductById(id));  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public ProductDataModel? GetElementByName(string name)  {  try  {  return \_mapper.Map<ProductDataModel>(\_dbContext.Products  .Include(x => x.Machine)  .Include(x => x.DetailProducts)!  .ThenInclude(x => x.Detail)  .FirstOrDefault(p => p.Name == name));  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void AddElement(ProductDataModel productDataModel)  {  try  {  \_dbContext.Products.Add(\_mapper.Map<Product>(productDataModel));  \_dbContext.SaveChanges();  }  catch (InvalidOperationException ex) when (ex.TargetSite?.Name == "ThrowIdentityConflict")  {  \_dbContext.ChangeTracker.Clear();  throw new ElementExistsException("Id", productDataModel.Id);  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void UpdElement(ProductDataModel productDataModel)  {  try  {  var existingProduct = GetProductById(productDataModel.Id)  ?? throw new ElementNotFoundException(productDataModel.Id);  \_dbContext.Products.Update(\_mapper.Map(productDataModel, existingProduct));  \_dbContext.SaveChanges();  }  catch (Exception ex) when (ex is ElementNotFoundException)  {  \_dbContext.ChangeTracker.Clear();  throw;  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void DelElement(string id)  {  try  {  var product = GetProductById(id) ?? throw new ElementNotFoundException(id);  \_dbContext.Products.Remove(product);  \_dbContext.SaveChanges();  }  catch (Exception ex) when (ex is ElementNotFoundException)  {  \_dbContext.ChangeTracker.Clear();  throw;  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  private Product? GetProductById(string id) => \_dbContext.Products  .Include(x => x.Machine)  .Include(x => x.DetailProducts)!  .ThenInclude(x => x.Detail)  .FirstOrDefault(p => p.Id == id);  }  // MachineStorageContract.cs  using AutoMapper;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.StoragesContracts;  using GoToWorkDatabase.Models;  using Microsoft.EntityFrameworkCore;  namespace GoToWorkDatabase.Implementations;  internal class MachineStorageContract : IMachineStorageContract  {  private readonly GoToWorkDbContext \_dbContext;  private readonly Mapper \_mapper;  public MachineStorageContract(GoToWorkDbContext dbContext)  {  \_dbContext = dbContext;  var config = new MapperConfiguration(cfg =>  {  cfg.CreateMap<Machine, MachineDataModel>()  .ForMember(dest => dest.Employees, opt => opt.MapFrom(src => src.EmployeeMachines))  .ForMember(dest => dest.Type, opt => opt.MapFrom(src => src.MachineType));  cfg.CreateMap<MachineDataModel, Machine>()  .ForMember(dest => dest.EmployeeMachines, opt => opt.MapFrom(src => src.Employees))  .ForMember(dest => dest.MachineType, opt => opt.MapFrom(src => src.Type));  cfg.CreateMap<Employee, EmployeeDataModel>();  cfg.CreateMap<EmployeeDataModel, Employee>();  cfg.CreateMap<EmployeeMachine, EmployeeMachineDataModel>();  cfg.CreateMap<EmployeeMachineDataModel, EmployeeMachine>();  cfg.CreateMap<Product, ProductDataModel>();  cfg.CreateMap<ProductDataModel, Product>();  });  \_mapper = new Mapper(config);  }  public List<MachineDataModel> GetList()  {  try  {  return  [  .. \_dbContext.Machines  .Include(x => x.Products)  .Include(x => x.EmployeeMachines)!  .ThenInclude(x => x.Employee)  .Select(m => \_mapper.Map<MachineDataModel>(m))  ];  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public MachineDataModel? GetElementById(string id)  {  try  {  return \_mapper.Map<MachineDataModel>(GetMachineById(id));  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public MachineDataModel? GetElementByModel(string model)  {  try  {  return \_mapper.Map<MachineDataModel>(\_dbContext.Machines  .Include(x => x.EmployeeMachines)!  .ThenInclude(x => x.Employee)  .Include(x => x.Products)  .FirstOrDefault(m => m.Model == model));  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void AddElement(MachineDataModel machineDataModel)  {  try  {  \_dbContext.Machines.Add(\_mapper.Map<Machine>(machineDataModel));  \_dbContext.SaveChanges();  }  catch (InvalidOperationException ex) when (ex.TargetSite?.Name == "ThrowIdentityConflict")  {  \_dbContext.ChangeTracker.Clear();  throw new ElementExistsException("Id", machineDataModel.Id);  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void UpdElement(MachineDataModel machineDataModel)  {  try  {  var existingMachine = GetMachineById(machineDataModel.Id)  ?? throw new ElementNotFoundException(machineDataModel.Id);  \_dbContext.Machines.Update(\_mapper.Map(machineDataModel, existingMachine));  \_dbContext.SaveChanges();  }  catch (Exception ex) when (ex is ElementNotFoundException)  {  \_dbContext.ChangeTracker.Clear();  throw;  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void DelElement(string id)  {  try  {  var machine = GetMachineById(id) ?? throw new ElementNotFoundException(id);  \_dbContext.Machines.Remove(machine);  \_dbContext.SaveChanges();  }  catch (Exception ex) when (ex is ElementNotFoundException)  {  \_dbContext.ChangeTracker.Clear();  throw;  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  private Machine? GetMachineById(string id)  {  return \_dbContext.Machines  .Include(x => x.EmployeeMachines)!  .ThenInclude(x => x.Employee)  .Include(x => x.Products)  .FirstOrDefault(m => m.Id == id);  }  }  // UserStorageContract.cs  using AutoMapper;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.StoragesContracts;  using GoToWorkDatabase.Models;  using Microsoft.EntityFrameworkCore;  namespace GoToWorkDatabase.Implementations;  internal class UserStorageContract : IUserStorageContract  {  private readonly GoToWorkDbContext \_dbContext;  private readonly Mapper \_mapper;  public UserStorageContract(GoToWorkDbContext dbContext)  {  \_dbContext = dbContext;  var config = new MapperConfiguration(cfg =>  {  cfg.CreateMap<User, UserDataModel>();  cfg.CreateMap<UserDataModel, User>();  });  \_mapper = new Mapper(config);  }  public List<UserDataModel> GetList()  {  try  {  return [.. \_dbContext.Users.Select(u => \_mapper.Map<UserDataModel>(u))];  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public UserDataModel? GetElementById(string id)  {  try  {  return \_mapper.Map<UserDataModel>(GetUserById(id));  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public UserDataModel? GetElementByLogin(string login)  {  try  {  return \_mapper.Map<UserDataModel>(\_dbContext.Users.FirstOrDefault(u => u.Login == login));  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public UserDataModel? GetElementByEmail(string email)  {  try  {  return \_mapper.Map<UserDataModel>(\_dbContext.Users.FirstOrDefault(u => u.Email == email));  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void AddElement(UserDataModel userDataModel)  {  try  {  \_dbContext.Users.Add(\_mapper.Map<User>(userDataModel));  \_dbContext.SaveChanges();  }  catch (InvalidOperationException ex) when (ex.TargetSite?.Name == "ThrowIdentityConflict")  {  \_dbContext.ChangeTracker.Clear();  throw new ElementExistsException("Id", userDataModel.Id);  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void UpdElement(UserDataModel userDataModel)  {  try  {  var existingUser = GetUserById(userDataModel.Id)  ?? throw new ElementNotFoundException(userDataModel.Id);  \_mapper.Map(userDataModel, existingUser);  \_dbContext.SaveChanges();  }  catch (Exception ex) when (ex is ElementNotFoundException)  {  \_dbContext.ChangeTracker.Clear();  throw;  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  public void DelElement(string id)  {  try  {  var user = GetUserById(id) ?? throw new ElementNotFoundException(id);  \_dbContext.Users.Remove(user);  \_dbContext.SaveChanges();  }  catch (Exception ex) when (ex is ElementNotFoundException or ElementDeletedException)  {  \_dbContext.ChangeTracker.Clear();  throw;  }  catch (Exception ex)  {  \_dbContext.ChangeTracker.Clear();  throw new StorageException(ex);  }  }  private User? GetUserById(string id) => \_dbContext.Users.FirstOrDefault(u => u.Id == id);  }  // User.cs  using GoToWorkContracts.Enums;  namespace GoToWorkDatabase.Models;  public class User  {  public required string Id { get; set; }  public required string Login { get; set; }  public required string Email { get; set; }  public required string Password { get; set; }  public required UserRole Role { get; set; }  public List<Employee>? Employees { get; set; }  }  // Production.cs  using System.ComponentModel.DataAnnotations.Schema;  namespace GoToWorkDatabase.Models;  public class Production  {  public string Id { get; set; } = Guid.NewGuid().ToString();  public required string Name { get; set; }  [ForeignKey("ProductionId")] public List<DetailProduction>? DetailProductions { get; set; }  [ForeignKey("ProductionId")] public List<Workshop>? Workshops { get; set; }  }  // Employee.cs  using System.ComponentModel.DataAnnotations.Schema;  namespace GoToWorkDatabase.Models;  public class Employee  {  public required string Id { get; set; }  public required string FullName { get; set; }  [ForeignKey("EmployeeId")] public List<EmployeeMachine>? EmployeeMachines { get; set; }  [ForeignKey("EmployeeId")] public List<EmployeeWorkshop>? EmployeeWorkshops { get; set; }  }  // Workshop.cs  using System.ComponentModel.DataAnnotations.Schema;  namespace GoToWorkDatabase.Models;  public class Workshop  {  public required string Id { get; set; }  public required string? ProductionId { get; set; }  public required string Address { get; set; }  public Production? Production { get; set; }  [ForeignKey("WorkshopId")] public List<EmployeeWorkshop>? EmployeeWorkshops { get; set; }  }  // DetailProduct.cs  namespace GoToWorkDatabase.Models;  public class DetailProduct  {  public required string ProductId { get; set; }  public required string DetailId { get; set; }  public required int Quantity { get; set; }  public Product? Product { get; set; }  public Detail? Detail { get; set; }  }  // EmployeeMachine.cs  namespace GoToWorkDatabase.Models;  public class EmployeeMachine  {  public required string EmployeeId { get; set; }  public required string MachineId { get; set; }  public Employee? Employee { get; set; }  public Machine? Machine { get; set; }  }  // DetailProduction.cs  namespace GoToWorkDatabase.Models;  public class DetailProduction  {  public required string DetailId { get; set; }  public required string ProductionId { get; set; }  public Detail? Detail { get; set; }  public Production? Production { get; set; }  }  // EmployeeWorkshop.cs  namespace GoToWorkDatabase.Models;  public class EmployeeWorkshop  {  public required string EmployeeId { get; set; }  public required string WorkshopId { get; set; }  public Employee? Employee { get; set; }  public Workshop? Workshop { get; set; }  }  // Machine.cs  using System.ComponentModel.DataAnnotations.Schema;  using GoToWorkContracts.Enums;  namespace GoToWorkDatabase.Models;  public class Machine  {  public required string Id { get; set; } = Guid.NewGuid().ToString();  public required string Model { get; set; }  public MachineType MachineType { get; set; }  [ForeignKey("MachineId")] public List<EmployeeMachine>? EmployeeMachines { get; set; }  [ForeignKey("MachineId")] public List<Product>? Products { get; set; }  }  // Product.cs  using System.ComponentModel.DataAnnotations.Schema;  namespace GoToWorkDatabase.Models;  public class Product  {  public required string Id { get; set; }  public required string? MachineId { get; set; }  public required string Name { get; set; }  public DateTime CreationDate { get; set; }  public Machine? Machine { get; set; }  [ForeignKey("ProductId")] public List<DetailProduct>? DetailProducts { get; set; }  }  // Detail.cs  using System.ComponentModel.DataAnnotations.Schema;  using GoToWorkContracts.Enums;  namespace GoToWorkDatabase.Models;  public class Detail  {  public required string Id { get; set; }  public required string Name { get; set; }  public MaterialType Material { get; set; }  public DateTime CreationDate { get; set; }  [ForeignKey("DetailId")] public List<DetailProduct>? DetailProducts { get; set; }  [ForeignKey("DetailId")] public List<DetailProduction>? DetailProductions { get; set; }  }  // GoToWorkDbContextModelSnapshot.cs  ﻿// <auto-generated />  using System;  using GoToWorkDatabase;  using Microsoft.EntityFrameworkCore;  using Microsoft.EntityFrameworkCore.Infrastructure;  using Microsoft.EntityFrameworkCore.Storage.ValueConversion;  using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;  #nullable disable  namespace GoToWorkDatabase.Migrations  {  [DbContext(typeof(GoToWorkDbContext))]  partial class GoToWorkDbContextModelSnapshot : ModelSnapshot  {  protected override void BuildModel(ModelBuilder modelBuilder)  {  #pragma warning disable 612, 618  modelBuilder  .HasAnnotation("ProductVersion", "9.0.4")  .HasAnnotation("Relational:MaxIdentifierLength", 63);  NpgsqlModelBuilderExtensions.UseIdentityByDefaultColumns(modelBuilder);  modelBuilder.Entity("GoToWorkDatabase.Models.Detail", b =>  {  b.Property<string>("Id")  .HasColumnType("text");  b.Property<DateTime>("CreationDate")  .HasColumnType("timestamp with time zone");  b.Property<int>("Material")  .HasColumnType("integer");  b.Property<string>("Name")  .IsRequired()  .HasColumnType("text");  b.HasKey("Id");  b.ToTable("Details");  });  modelBuilder.Entity("GoToWorkDatabase.Models.DetailProduct", b =>  {  b.Property<string>("DetailId")  .HasColumnType("text");  b.Property<string>("ProductId")  .HasColumnType("text");  b.Property<int>("Quantity")  .HasColumnType("integer");  b.HasKey("DetailId", "ProductId");  b.HasIndex("ProductId");  b.ToTable("DetailProducts");  });  modelBuilder.Entity("GoToWorkDatabase.Models.DetailProduction", b =>  {  b.Property<string>("DetailId")  .HasColumnType("text");  b.Property<string>("ProductionId")  .HasColumnType("text");  b.HasKey("DetailId", "ProductionId");  b.HasIndex("ProductionId");  b.ToTable("DetailProductions");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Employee", b =>  {  b.Property<string>("Id")  .HasColumnType("text");  b.Property<string>("FullName")  .IsRequired()  .HasColumnType("text");  b.Property<string>("UserId")  .HasColumnType("text");  b.HasKey("Id");  b.HasIndex("UserId");  b.ToTable("Employees");  });  modelBuilder.Entity("GoToWorkDatabase.Models.EmployeeMachine", b =>  {  b.Property<string>("EmployeeId")  .HasColumnType("text");  b.Property<string>("MachineId")  .HasColumnType("text");  b.HasKey("EmployeeId", "MachineId");  b.HasIndex("MachineId");  b.ToTable("EmployeeMachines");  });  modelBuilder.Entity("GoToWorkDatabase.Models.EmployeeWorkshop", b =>  {  b.Property<string>("EmployeeId")  .HasColumnType("text");  b.Property<string>("WorkshopId")  .HasColumnType("text");  b.HasKey("EmployeeId", "WorkshopId");  b.HasIndex("WorkshopId");  b.ToTable("EmployeeWorkshops");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Machine", b =>  {  b.Property<string>("Id")  .HasColumnType("text");  b.Property<int>("MachineType")  .HasColumnType("integer");  b.Property<string>("Model")  .IsRequired()  .HasColumnType("text");  b.HasKey("Id");  b.HasIndex("Model")  .IsUnique();  b.ToTable("Machines");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Product", b =>  {  b.Property<string>("Id")  .HasColumnType("text");  b.Property<DateTime>("CreationDate")  .HasColumnType("timestamp with time zone");  b.Property<string>("MachineId")  .HasColumnType("text");  b.Property<string>("Name")  .IsRequired()  .HasColumnType("text");  b.HasKey("Id");  b.HasIndex("MachineId");  b.HasIndex("Name")  .IsUnique();  b.ToTable("Products");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Production", b =>  {  b.Property<string>("Id")  .HasColumnType("text");  b.Property<string>("Name")  .IsRequired()  .HasColumnType("text");  b.HasKey("Id");  b.HasIndex("Name")  .IsUnique();  b.ToTable("Productions");  });  modelBuilder.Entity("GoToWorkDatabase.Models.User", b =>  {  b.Property<string>("Id")  .HasColumnType("text");  b.Property<string>("Email")  .IsRequired()  .HasColumnType("text");  b.Property<string>("Login")  .IsRequired()  .HasColumnType("text");  b.Property<string>("Password")  .IsRequired()  .HasColumnType("text");  b.Property<int>("Role")  .HasColumnType("integer");  b.HasKey("Id");  b.HasIndex("Email")  .IsUnique();  b.HasIndex("Login")  .IsUnique();  b.ToTable("Users");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Workshop", b =>  {  b.Property<string>("Id")  .HasColumnType("text");  b.Property<string>("Address")  .IsRequired()  .HasColumnType("text");  b.Property<string>("ProductionId")  .HasColumnType("text");  b.HasKey("Id");  b.HasIndex("Address")  .IsUnique();  b.HasIndex("ProductionId");  b.ToTable("Workshops");  });  modelBuilder.Entity("GoToWorkDatabase.Models.DetailProduct", b =>  {  b.HasOne("GoToWorkDatabase.Models.Detail", "Detail")  .WithMany("DetailProducts")  .HasForeignKey("DetailId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.HasOne("GoToWorkDatabase.Models.Product", "Product")  .WithMany("DetailProducts")  .HasForeignKey("ProductId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.Navigation("Detail");  b.Navigation("Product");  });  modelBuilder.Entity("GoToWorkDatabase.Models.DetailProduction", b =>  {  b.HasOne("GoToWorkDatabase.Models.Detail", "Detail")  .WithMany("DetailProductions")  .HasForeignKey("DetailId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.HasOne("GoToWorkDatabase.Models.Production", "Production")  .WithMany("DetailProductions")  .HasForeignKey("ProductionId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.Navigation("Detail");  b.Navigation("Production");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Employee", b =>  {  b.HasOne("GoToWorkDatabase.Models.User", null)  .WithMany("Employees")  .HasForeignKey("UserId");  });  modelBuilder.Entity("GoToWorkDatabase.Models.EmployeeMachine", b =>  {  b.HasOne("GoToWorkDatabase.Models.Employee", "Employee")  .WithMany("EmployeeMachines")  .HasForeignKey("EmployeeId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.HasOne("GoToWorkDatabase.Models.Machine", "Machine")  .WithMany("EmployeeMachines")  .HasForeignKey("MachineId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.Navigation("Employee");  b.Navigation("Machine");  });  modelBuilder.Entity("GoToWorkDatabase.Models.EmployeeWorkshop", b =>  {  b.HasOne("GoToWorkDatabase.Models.Employee", "Employee")  .WithMany("EmployeeWorkshops")  .HasForeignKey("EmployeeId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.HasOne("GoToWorkDatabase.Models.Workshop", "Workshop")  .WithMany("EmployeeWorkshops")  .HasForeignKey("WorkshopId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.Navigation("Employee");  b.Navigation("Workshop");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Product", b =>  {  b.HasOne("GoToWorkDatabase.Models.Machine", "Machine")  .WithMany("Products")  .HasForeignKey("MachineId");  b.Navigation("Machine");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Workshop", b =>  {  b.HasOne("GoToWorkDatabase.Models.Production", "Production")  .WithMany("Workshops")  .HasForeignKey("ProductionId");  b.Navigation("Production");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Detail", b =>  {  b.Navigation("DetailProductions");  b.Navigation("DetailProducts");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Employee", b =>  {  b.Navigation("EmployeeMachines");  b.Navigation("EmployeeWorkshops");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Machine", b =>  {  b.Navigation("EmployeeMachines");  b.Navigation("Products");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Product", b =>  {  b.Navigation("DetailProducts");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Production", b =>  {  b.Navigation("DetailProductions");  b.Navigation("Workshops");  });  modelBuilder.Entity("GoToWorkDatabase.Models.User", b =>  {  b.Navigation("Employees");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Workshop", b =>  {  b.Navigation("EmployeeWorkshops");  });  #pragma warning restore 612, 618  }  }  }  // 20250825165651\_Initial.cs  ﻿using System;  using Microsoft.EntityFrameworkCore.Migrations;  #nullable disable  namespace GoToWorkDatabase.Migrations  {  /// <inheritdoc />  public partial class Initial : Migration  {  /// <inheritdoc />  protected override void Up(MigrationBuilder migrationBuilder)  {  migrationBuilder.CreateTable(  name: "Details",  columns: table => new  {  Id = table.Column<string>(type: "text", nullable: false),  Name = table.Column<string>(type: "text", nullable: false),  Material = table.Column<int>(type: "integer", nullable: false),  CreationDate = table.Column<DateTime>(type: "timestamp with time zone", nullable: false)  },  constraints: table =>  {  table.PrimaryKey("PK\_Details", x => x.Id);  });  migrationBuilder.CreateTable(  name: "Machines",  columns: table => new  {  Id = table.Column<string>(type: "text", nullable: false),  Model = table.Column<string>(type: "text", nullable: false),  MachineType = table.Column<int>(type: "integer", nullable: false)  },  constraints: table =>  {  table.PrimaryKey("PK\_Machines", x => x.Id);  });  migrationBuilder.CreateTable(  name: "Productions",  columns: table => new  {  Id = table.Column<string>(type: "text", nullable: false),  Name = table.Column<string>(type: "text", nullable: false)  },  constraints: table =>  {  table.PrimaryKey("PK\_Productions", x => x.Id);  });  migrationBuilder.CreateTable(  name: "Users",  columns: table => new  {  Id = table.Column<string>(type: "text", nullable: false),  Login = table.Column<string>(type: "text", nullable: false),  Email = table.Column<string>(type: "text", nullable: false),  Password = table.Column<string>(type: "text", nullable: false),  Role = table.Column<int>(type: "integer", nullable: false)  },  constraints: table =>  {  table.PrimaryKey("PK\_Users", x => x.Id);  });  migrationBuilder.CreateTable(  name: "Products",  columns: table => new  {  Id = table.Column<string>(type: "text", nullable: false),  MachineId = table.Column<string>(type: "text", nullable: true),  Name = table.Column<string>(type: "text", nullable: false),  CreationDate = table.Column<DateTime>(type: "timestamp with time zone", nullable: false)  },  constraints: table =>  {  table.PrimaryKey("PK\_Products", x => x.Id);  table.ForeignKey(  name: "FK\_Products\_Machines\_MachineId",  column: x => x.MachineId,  principalTable: "Machines",  principalColumn: "Id");  });  migrationBuilder.CreateTable(  name: "DetailProductions",  columns: table => new  {  DetailId = table.Column<string>(type: "text", nullable: false),  ProductionId = table.Column<string>(type: "text", nullable: false)  },  constraints: table =>  {  table.PrimaryKey("PK\_DetailProductions", x => new { x.DetailId, x.ProductionId });  table.ForeignKey(  name: "FK\_DetailProductions\_Details\_DetailId",  column: x => x.DetailId,  principalTable: "Details",  principalColumn: "Id",  onDelete: ReferentialAction.Cascade);  table.ForeignKey(  name: "FK\_DetailProductions\_Productions\_ProductionId",  column: x => x.ProductionId,  principalTable: "Productions",  principalColumn: "Id",  onDelete: ReferentialAction.Cascade);  });  migrationBuilder.CreateTable(  name: "Workshops",  columns: table => new  {  Id = table.Column<string>(type: "text", nullable: false),  ProductionId = table.Column<string>(type: "text", nullable: true),  Address = table.Column<string>(type: "text", nullable: false)  },  constraints: table =>  {  table.PrimaryKey("PK\_Workshops", x => x.Id);  table.ForeignKey(  name: "FK\_Workshops\_Productions\_ProductionId",  column: x => x.ProductionId,  principalTable: "Productions",  principalColumn: "Id");  });  migrationBuilder.CreateTable(  name: "Employees",  columns: table => new  {  Id = table.Column<string>(type: "text", nullable: false),  FullName = table.Column<string>(type: "text", nullable: false),  UserId = table.Column<string>(type: "text", nullable: true)  },  constraints: table =>  {  table.PrimaryKey("PK\_Employees", x => x.Id);  table.ForeignKey(  name: "FK\_Employees\_Users\_UserId",  column: x => x.UserId,  principalTable: "Users",  principalColumn: "Id");  });  migrationBuilder.CreateTable(  name: "DetailProducts",  columns: table => new  {  ProductId = table.Column<string>(type: "text", nullable: false),  DetailId = table.Column<string>(type: "text", nullable: false),  Quantity = table.Column<int>(type: "integer", nullable: false)  },  constraints: table =>  {  table.PrimaryKey("PK\_DetailProducts", x => new { x.DetailId, x.ProductId });  table.ForeignKey(  name: "FK\_DetailProducts\_Details\_DetailId",  column: x => x.DetailId,  principalTable: "Details",  principalColumn: "Id",  onDelete: ReferentialAction.Cascade);  table.ForeignKey(  name: "FK\_DetailProducts\_Products\_ProductId",  column: x => x.ProductId,  principalTable: "Products",  principalColumn: "Id",  onDelete: ReferentialAction.Cascade);  });  migrationBuilder.CreateTable(  name: "EmployeeMachines",  columns: table => new  {  EmployeeId = table.Column<string>(type: "text", nullable: false),  MachineId = table.Column<string>(type: "text", nullable: false)  },  constraints: table =>  {  table.PrimaryKey("PK\_EmployeeMachines", x => new { x.EmployeeId, x.MachineId });  table.ForeignKey(  name: "FK\_EmployeeMachines\_Employees\_EmployeeId",  column: x => x.EmployeeId,  principalTable: "Employees",  principalColumn: "Id",  onDelete: ReferentialAction.Cascade);  table.ForeignKey(  name: "FK\_EmployeeMachines\_Machines\_MachineId",  column: x => x.MachineId,  principalTable: "Machines",  principalColumn: "Id",  onDelete: ReferentialAction.Cascade);  });  migrationBuilder.CreateTable(  name: "EmployeeWorkshops",  columns: table => new  {  EmployeeId = table.Column<string>(type: "text", nullable: false),  WorkshopId = table.Column<string>(type: "text", nullable: false)  },  constraints: table =>  {  table.PrimaryKey("PK\_EmployeeWorkshops", x => new { x.EmployeeId, x.WorkshopId });  table.ForeignKey(  name: "FK\_EmployeeWorkshops\_Employees\_EmployeeId",  column: x => x.EmployeeId,  principalTable: "Employees",  principalColumn: "Id",  onDelete: ReferentialAction.Cascade);  table.ForeignKey(  name: "FK\_EmployeeWorkshops\_Workshops\_WorkshopId",  column: x => x.WorkshopId,  principalTable: "Workshops",  principalColumn: "Id",  onDelete: ReferentialAction.Cascade);  });  migrationBuilder.CreateIndex(  name: "IX\_DetailProductions\_ProductionId",  table: "DetailProductions",  column: "ProductionId");  migrationBuilder.CreateIndex(  name: "IX\_DetailProducts\_ProductId",  table: "DetailProducts",  column: "ProductId");  migrationBuilder.CreateIndex(  name: "IX\_EmployeeMachines\_MachineId",  table: "EmployeeMachines",  column: "MachineId");  migrationBuilder.CreateIndex(  name: "IX\_Employees\_UserId",  table: "Employees",  column: "UserId");  migrationBuilder.CreateIndex(  name: "IX\_EmployeeWorkshops\_WorkshopId",  table: "EmployeeWorkshops",  column: "WorkshopId");  migrationBuilder.CreateIndex(  name: "IX\_Machines\_Model",  table: "Machines",  column: "Model",  unique: true);  migrationBuilder.CreateIndex(  name: "IX\_Productions\_Name",  table: "Productions",  column: "Name",  unique: true);  migrationBuilder.CreateIndex(  name: "IX\_Products\_MachineId",  table: "Products",  column: "MachineId");  migrationBuilder.CreateIndex(  name: "IX\_Products\_Name",  table: "Products",  column: "Name",  unique: true);  migrationBuilder.CreateIndex(  name: "IX\_Users\_Email",  table: "Users",  column: "Email",  unique: true);  migrationBuilder.CreateIndex(  name: "IX\_Users\_Login",  table: "Users",  column: "Login",  unique: true);  migrationBuilder.CreateIndex(  name: "IX\_Workshops\_Address",  table: "Workshops",  column: "Address",  unique: true);  migrationBuilder.CreateIndex(  name: "IX\_Workshops\_ProductionId",  table: "Workshops",  column: "ProductionId");  }  /// <inheritdoc />  protected override void Down(MigrationBuilder migrationBuilder)  {  migrationBuilder.DropTable(  name: "DetailProductions");  migrationBuilder.DropTable(  name: "DetailProducts");  migrationBuilder.DropTable(  name: "EmployeeMachines");  migrationBuilder.DropTable(  name: "EmployeeWorkshops");  migrationBuilder.DropTable(  name: "Details");  migrationBuilder.DropTable(  name: "Products");  migrationBuilder.DropTable(  name: "Employees");  migrationBuilder.DropTable(  name: "Workshops");  migrationBuilder.DropTable(  name: "Machines");  migrationBuilder.DropTable(  name: "Users");  migrationBuilder.DropTable(  name: "Productions");  }  }  }  // 20250825165651\_Initial.Designer.cs  ﻿// <auto-generated />  using System;  using GoToWorkDatabase;  using Microsoft.EntityFrameworkCore;  using Microsoft.EntityFrameworkCore.Infrastructure;  using Microsoft.EntityFrameworkCore.Migrations;  using Microsoft.EntityFrameworkCore.Storage.ValueConversion;  using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;  #nullable disable  namespace GoToWorkDatabase.Migrations  {  [DbContext(typeof(GoToWorkDbContext))]  [Migration("20250825165651\_Initial")]  partial class Initial  {  /// <inheritdoc />  protected override void BuildTargetModel(ModelBuilder modelBuilder)  {  #pragma warning disable 612, 618  modelBuilder  .HasAnnotation("ProductVersion", "9.0.4")  .HasAnnotation("Relational:MaxIdentifierLength", 63);  NpgsqlModelBuilderExtensions.UseIdentityByDefaultColumns(modelBuilder);  modelBuilder.Entity("GoToWorkDatabase.Models.Detail", b =>  {  b.Property<string>("Id")  .HasColumnType("text");  b.Property<DateTime>("CreationDate")  .HasColumnType("timestamp with time zone");  b.Property<int>("Material")  .HasColumnType("integer");  b.Property<string>("Name")  .IsRequired()  .HasColumnType("text");  b.HasKey("Id");  b.ToTable("Details");  });  modelBuilder.Entity("GoToWorkDatabase.Models.DetailProduct", b =>  {  b.Property<string>("DetailId")  .HasColumnType("text");  b.Property<string>("ProductId")  .HasColumnType("text");  b.Property<int>("Quantity")  .HasColumnType("integer");  b.HasKey("DetailId", "ProductId");  b.HasIndex("ProductId");  b.ToTable("DetailProducts");  });  modelBuilder.Entity("GoToWorkDatabase.Models.DetailProduction", b =>  {  b.Property<string>("DetailId")  .HasColumnType("text");  b.Property<string>("ProductionId")  .HasColumnType("text");  b.HasKey("DetailId", "ProductionId");  b.HasIndex("ProductionId");  b.ToTable("DetailProductions");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Employee", b =>  {  b.Property<string>("Id")  .HasColumnType("text");  b.Property<string>("FullName")  .IsRequired()  .HasColumnType("text");  b.Property<string>("UserId")  .HasColumnType("text");  b.HasKey("Id");  b.HasIndex("UserId");  b.ToTable("Employees");  });  modelBuilder.Entity("GoToWorkDatabase.Models.EmployeeMachine", b =>  {  b.Property<string>("EmployeeId")  .HasColumnType("text");  b.Property<string>("MachineId")  .HasColumnType("text");  b.HasKey("EmployeeId", "MachineId");  b.HasIndex("MachineId");  b.ToTable("EmployeeMachines");  });  modelBuilder.Entity("GoToWorkDatabase.Models.EmployeeWorkshop", b =>  {  b.Property<string>("EmployeeId")  .HasColumnType("text");  b.Property<string>("WorkshopId")  .HasColumnType("text");  b.HasKey("EmployeeId", "WorkshopId");  b.HasIndex("WorkshopId");  b.ToTable("EmployeeWorkshops");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Machine", b =>  {  b.Property<string>("Id")  .HasColumnType("text");  b.Property<int>("MachineType")  .HasColumnType("integer");  b.Property<string>("Model")  .IsRequired()  .HasColumnType("text");  b.HasKey("Id");  b.HasIndex("Model")  .IsUnique();  b.ToTable("Machines");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Product", b =>  {  b.Property<string>("Id")  .HasColumnType("text");  b.Property<DateTime>("CreationDate")  .HasColumnType("timestamp with time zone");  b.Property<string>("MachineId")  .HasColumnType("text");  b.Property<string>("Name")  .IsRequired()  .HasColumnType("text");  b.HasKey("Id");  b.HasIndex("MachineId");  b.HasIndex("Name")  .IsUnique();  b.ToTable("Products");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Production", b =>  {  b.Property<string>("Id")  .HasColumnType("text");  b.Property<string>("Name")  .IsRequired()  .HasColumnType("text");  b.HasKey("Id");  b.HasIndex("Name")  .IsUnique();  b.ToTable("Productions");  });  modelBuilder.Entity("GoToWorkDatabase.Models.User", b =>  {  b.Property<string>("Id")  .HasColumnType("text");  b.Property<string>("Email")  .IsRequired()  .HasColumnType("text");  b.Property<string>("Login")  .IsRequired()  .HasColumnType("text");  b.Property<string>("Password")  .IsRequired()  .HasColumnType("text");  b.Property<int>("Role")  .HasColumnType("integer");  b.HasKey("Id");  b.HasIndex("Email")  .IsUnique();  b.HasIndex("Login")  .IsUnique();  b.ToTable("Users");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Workshop", b =>  {  b.Property<string>("Id")  .HasColumnType("text");  b.Property<string>("Address")  .IsRequired()  .HasColumnType("text");  b.Property<string>("ProductionId")  .HasColumnType("text");  b.HasKey("Id");  b.HasIndex("Address")  .IsUnique();  b.HasIndex("ProductionId");  b.ToTable("Workshops");  });  modelBuilder.Entity("GoToWorkDatabase.Models.DetailProduct", b =>  {  b.HasOne("GoToWorkDatabase.Models.Detail", "Detail")  .WithMany("DetailProducts")  .HasForeignKey("DetailId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.HasOne("GoToWorkDatabase.Models.Product", "Product")  .WithMany("DetailProducts")  .HasForeignKey("ProductId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.Navigation("Detail");  b.Navigation("Product");  });  modelBuilder.Entity("GoToWorkDatabase.Models.DetailProduction", b =>  {  b.HasOne("GoToWorkDatabase.Models.Detail", "Detail")  .WithMany("DetailProductions")  .HasForeignKey("DetailId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.HasOne("GoToWorkDatabase.Models.Production", "Production")  .WithMany("DetailProductions")  .HasForeignKey("ProductionId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.Navigation("Detail");  b.Navigation("Production");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Employee", b =>  {  b.HasOne("GoToWorkDatabase.Models.User", null)  .WithMany("Employees")  .HasForeignKey("UserId");  });  modelBuilder.Entity("GoToWorkDatabase.Models.EmployeeMachine", b =>  {  b.HasOne("GoToWorkDatabase.Models.Employee", "Employee")  .WithMany("EmployeeMachines")  .HasForeignKey("EmployeeId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.HasOne("GoToWorkDatabase.Models.Machine", "Machine")  .WithMany("EmployeeMachines")  .HasForeignKey("MachineId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.Navigation("Employee");  b.Navigation("Machine");  });  modelBuilder.Entity("GoToWorkDatabase.Models.EmployeeWorkshop", b =>  {  b.HasOne("GoToWorkDatabase.Models.Employee", "Employee")  .WithMany("EmployeeWorkshops")  .HasForeignKey("EmployeeId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.HasOne("GoToWorkDatabase.Models.Workshop", "Workshop")  .WithMany("EmployeeWorkshops")  .HasForeignKey("WorkshopId")  .OnDelete(DeleteBehavior.Cascade)  .IsRequired();  b.Navigation("Employee");  b.Navigation("Workshop");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Product", b =>  {  b.HasOne("GoToWorkDatabase.Models.Machine", "Machine")  .WithMany("Products")  .HasForeignKey("MachineId");  b.Navigation("Machine");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Workshop", b =>  {  b.HasOne("GoToWorkDatabase.Models.Production", "Production")  .WithMany("Workshops")  .HasForeignKey("ProductionId");  b.Navigation("Production");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Detail", b =>  {  b.Navigation("DetailProductions");  b.Navigation("DetailProducts");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Employee", b =>  {  b.Navigation("EmployeeMachines");  b.Navigation("EmployeeWorkshops");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Machine", b =>  {  b.Navigation("EmployeeMachines");  b.Navigation("Products");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Product", b =>  {  b.Navigation("DetailProducts");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Production", b =>  {  b.Navigation("DetailProductions");  b.Navigation("Workshops");  });  modelBuilder.Entity("GoToWorkDatabase.Models.User", b =>  {  b.Navigation("Employees");  });  modelBuilder.Entity("GoToWorkDatabase.Models.Workshop", b =>  {  b.Navigation("EmployeeWorkshops");  });  #pragma warning restore 612, 618  }  }  }  // GoToWorkDatabase.csproj.nuget.g.targets  ﻿<?xml version="1.0" encoding="utf-8" standalone="no"?>  <Project ToolsVersion="14.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">  <ImportGroup Condition=" '$(ExcludeRestorePackageImports)' != 'true' ">  <Import Project="$(NuGetPackageRoot)system.text.json/9.0.4/buildTransitive/net8.0/System.Text.Json.targets" Condition="Exists('$(NuGetPackageRoot)system.text.json/9.0.4/buildTransitive/net8.0/System.Text.Json.targets')" />  <Import Project="$(NuGetPackageRoot)microsoft.extensions.logging.abstractions/9.0.4/buildTransitive/net8.0/Microsoft.Extensions.Logging.Abstractions.targets" Condition="Exists('$(NuGetPackageRoot)microsoft.extensions.logging.abstractions/9.0.4/buildTransitive/net8.0/Microsoft.Extensions.Logging.Abstractions.targets')" />  <Import Project="$(NuGetPackageRoot)microsoft.extensions.options/9.0.4/buildTransitive/net8.0/Microsoft.Extensions.Options.targets" Condition="Exists('$(NuGetPackageRoot)microsoft.extensions.options/9.0.4/buildTransitive/net8.0/Microsoft.Extensions.Options.targets')" />  <Import Project="$(NuGetPackageRoot)mono.texttemplating/3.0.0/buildTransitive/Mono.TextTemplating.targets" Condition="Exists('$(NuGetPackageRoot)mono.texttemplating/3.0.0/buildTransitive/Mono.TextTemplating.targets')" />  <Import Project="$(NuGetPackageRoot)microsoft.codeanalysis.analyzers/3.3.4/buildTransitive/Microsoft.CodeAnalysis.Analyzers.targets" Condition="Exists('$(NuGetPackageRoot)microsoft.codeanalysis.analyzers/3.3.4/buildTransitive/Microsoft.CodeAnalysis.Analyzers.targets')" />  </ImportGroup>  </Project>  // GoToWorkDatabase.csproj.EntityFrameworkCore.targets  ﻿<?xml version="1.0" encoding="utf-8"?>  <Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">  <Target Name="GetEFProjectMetadata">  <MSBuild Condition=" '$(TargetFramework)' == '' "  Projects="$(MSBuildProjectFile)"  Targets="GetEFProjectMetadata"  Properties="TargetFramework=$(TargetFrameworks.Split(';')[0]);EFProjectMetadataFile=$(EFProjectMetadataFile)" />  <ItemGroup Condition=" '$(TargetFramework)' != '' ">  <EFProjectMetadata Include="AssemblyName: $(AssemblyName)" />  <EFProjectMetadata Include="Language: $(Language)" />  <EFProjectMetadata Include="OutputPath: $(OutputPath)" />  <EFProjectMetadata Include="Platform: $(Platform)" />  <EFProjectMetadata Include="PlatformTarget: $(PlatformTarget)" />  <EFProjectMetadata Include="ProjectAssetsFile: $(ProjectAssetsFile)" />  <EFProjectMetadata Include="ProjectDir: $(ProjectDir)" />  <EFProjectMetadata Include="RootNamespace: $(RootNamespace)" />  <EFProjectMetadata Include="RuntimeFrameworkVersion: $(RuntimeFrameworkVersion)" />  <EFProjectMetadata Include="TargetFileName: $(TargetFileName)" />  <EFProjectMetadata Include="TargetFrameworkMoniker: $(TargetFrameworkMoniker)" />  <EFProjectMetadata Include="Nullable: $(Nullable)" />  <EFProjectMetadata Include="TargetFramework: $(TargetFramework)" />  <EFProjectMetadata Include="TargetPlatformIdentifier: $(TargetPlatformIdentifier)" />  </ItemGroup>  <WriteLinesToFile Condition=" '$(TargetFramework)' != '' "  File="$(EFProjectMetadataFile)"  Lines="@(EFProjectMetadata)" />  </Target>  </Project>  // GoToWorkDatabase.GlobalUsings.g.cs  // <auto-generated/>  global using global::System;  global using global::System.Collections.Generic;  global using global::System.IO;  global using global::System.Linq;  global using global::System.Net.Http;  global using global::System.Threading;  global using global::System.Threading.Tasks;  // .NETCoreApp,Version=v9.0.AssemblyAttributes.cs  // <autogenerated />  using System;  using System.Reflection;  [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v9.0", FrameworkDisplayName = ".NET 9.0")]  // GoToWorkDatabase.AssemblyInfo.cs  //------------------------------------------------------------------------------  // <auto-generated>  // This code was generated by a tool.  //  // Changes to this file may cause incorrect behavior and will be lost if  // the code is regenerated.  // </auto-generated>  //------------------------------------------------------------------------------  using System;  using System.Reflection;  [assembly: System.Reflection.AssemblyCompanyAttribute("GoToWorkDatabase")]  [assembly: System.Reflection.AssemblyConfigurationAttribute("Release")]  [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]  [assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+8753cd23468d9b41857b154ffbb5889134ac4e4c")]  [assembly: System.Reflection.AssemblyProductAttribute("GoToWorkDatabase")]  [assembly: System.Reflection.AssemblyTitleAttribute("GoToWorkDatabase")]  [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("GoToWorkApi")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("GoToWorkBusinessLogic")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("GoToWorkContracts")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("DynamicProxyGenAssembly2")]  // Generated by the MSBuild WriteCodeFragment class.  // GoToWorkDatabase.GlobalUsings.g.cs  // <auto-generated/>  global using global::System;  global using global::System.Collections.Generic;  global using global::System.IO;  global using global::System.Linq;  global using global::System.Net.Http;  global using global::System.Threading;  global using global::System.Threading.Tasks;  // .NETCoreApp,Version=v9.0.AssemblyAttributes.cs  // <autogenerated />  using System;  using System.Reflection;  [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v9.0", FrameworkDisplayName = ".NET 9.0")]  // GoToWorkDatabase.AssemblyInfo.cs  //------------------------------------------------------------------------------  // <auto-generated>  // This code was generated by a tool.  //  // Changes to this file may cause incorrect behavior and will be lost if  // the code is regenerated.  // </auto-generated>  //------------------------------------------------------------------------------  using System;  using System.Reflection;  [assembly: System.Reflection.AssemblyCompanyAttribute("GoToWorkDatabase")]  [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]  [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]  [assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+8753cd23468d9b41857b154ffbb5889134ac4e4c")]  [assembly: System.Reflection.AssemblyProductAttribute("GoToWorkDatabase")]  [assembly: System.Reflection.AssemblyTitleAttribute("GoToWorkDatabase")]  [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("GoToWorkApi")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("GoToWorkBusinessLogic")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("GoToWorkContracts")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("DynamicProxyGenAssembly2")]  // Generated by the MSBuild WriteCodeFragment class.  // ProductBusinessLogicContract.cs  using System.Text.Json;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.StoragesContracts;  using Microsoft.Extensions.Logging;  namespace GoToWorkBusinessLogic.Implementations;  public class ProductBusinessLogicContract(  IProductStorageContract productStorageContract,  ILogger logger) : IProductBusinessLogicContract  {  public List<ProductDataModel> GetAllProducts()  {  logger.LogInformation("Getting all products");  return productStorageContract.GetList() ?? throw new NullListException();  }  public List<ProductDataModel> GetProductsByMachine(string machineId)  {  logger.LogInformation("Getting products by machine: {machineId}", machineId);  if (machineId.IsEmpty()) throw new ArgumentNullException(nameof(machineId));  if (!machineId.IsGuid()) throw new ValidationException("MachineId is not a unique identifier");  return productStorageContract.GetList(machineId: machineId) ?? throw new NullListException();  }  public List<ProductDataModel> GetProductsByCreationDate(DateTime from, DateTime to)  {  logger.LogInformation("Getting products by date range: {from} - {to}", from, to);  if (from > to) throw new ValidationException("Start date cannot be later than end date");  return productStorageContract.GetList(from, to) ?? throw new NullListException();  }  public ProductDataModel GetProductByData(string data)  {  logger.LogInformation("Getting product by data: {data}", data);  if (data.IsEmpty()) throw new ArgumentNullException(nameof(data));  if (data.IsGuid())  return productStorageContract.GetElementById(data) ?? throw new ElementNotFoundException(data);  return productStorageContract.GetElementByName(data) ?? throw new ElementNotFoundException(data);  }  public void InsertProduct(ProductDataModel product)  {  logger.LogInformation("Inserting new product: {json}", JsonSerializer.Serialize(product));  ArgumentNullException.ThrowIfNull(product);  product.Validate();  productStorageContract.AddElement(product);  }  public void UpdateProduct(ProductDataModel product)  {  logger.LogInformation("Updating product: {json}", JsonSerializer.Serialize(product));  ArgumentNullException.ThrowIfNull(product);  product.Validate();  productStorageContract.UpdElement(product);  }  public void DeleteProduct(string id)  {  logger.LogInformation("Deleting product with id: {id}", id);  if (id.IsEmpty()) throw new ArgumentNullException(nameof(id));  if (!id.IsGuid()) throw new ValidationException("Id is not a unique identifier");  productStorageContract.DelElement(id);  }  }  // ReportContract.cs  using System.Diagnostics;  using System.Net;  using System.Net.Mail;  using System.Text;  using ClosedXML.Excel;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.ViewModels;  using GoToWorkDatabase;  using Microsoft.EntityFrameworkCore;  using Microsoft.Extensions.Logging;  using Xceed.Words.NET;  namespace GoToWorkBusinessLogic.Implementations;  public class ReportContract(  GoToWorkDbContext context,  ILogger logger)  : IReportContract  {  public async Task<List<WorkshopsReportViewModel>> GetWorkshopsByDetailsAsync(List<string> selectedDetailIds,  CancellationToken ct)  {  return await context.Workshops  .Include(w => w.Production)  .ThenInclude(p => p!.DetailProductions)!  .ThenInclude(dp => dp.Detail)  .Where(w => w.Production!.DetailProductions!  .Any(dp => selectedDetailIds.Contains(dp.DetailId)))  .Select(w => new WorkshopsReportViewModel  {  WorkshopId = w.Id,  Address = w.Address,  ProductionName = w.Production!.Name,  RelatedDetails = w.Production.DetailProductions!  .Where(dp => selectedDetailIds.Contains(dp.DetailId))  .Select(dp => dp.Detail!.Name)  .ToList()  })  .ToListAsync(ct);  }  public async Task<Stream> CreateDocxDocumentWorkshopsByDetailsAsync(List<WorkshopsReportViewModel> data,  CancellationToken ct)  {  var stream = new MemoryStream();  using var doc = DocX.Create(stream);  var table = doc.InsertTable(data.Count + 1, 4);  // Заголовки  table.Rows[0].Cells[0].Paragraphs.First().Append("Id Цеха");  table.Rows[0].Cells[1].Paragraphs.First().Append("Адрес");  table.Rows[0].Cells[2].Paragraphs.First().Append("Название производства");  table.Rows[0].Cells[3].Paragraphs.First().Append("Связанные детали");  // Данные  for (var i = 0; i < data.Count; i++)  {  table.Rows[i + 1].Cells[0].Paragraphs.First().Append(data[i].WorkshopId);  table.Rows[i + 1].Cells[1].Paragraphs.First().Append(data[i].Address);  table.Rows[i + 1].Cells[2].Paragraphs.First().Append(data[i].ProductionName);  table.Rows[i + 1].Cells[3].Paragraphs.First().Append(string.Join(", ", data[i].RelatedDetails));  }  doc.Save();  stream.Position = 0;  return stream;  }  public async Task<Stream> CreateXlsxDocumentWorkshopsByDetailsAsync(List<WorkshopsReportViewModel> data,  CancellationToken ct)  {  using var workbook = new XLWorkbook();  var worksheet = workbook.Worksheets.Add("Workshops");  // Заголовки  worksheet.Cell(1, 1).Value = "Id Цеха";  worksheet.Cell(1, 2).Value = "Адрес";  worksheet.Cell(1, 3).Value = "Название производства";  worksheet.Cell(1, 4).Value = "Связанные детали";  // Данные  for (var i = 0; i < data.Count; i++)  {  worksheet.Cell(i + 2, 1).Value = data[i].WorkshopId;  worksheet.Cell(i + 2, 2).Value = data[i].Address;  worksheet.Cell(i + 2, 3).Value = data[i].ProductionName;  worksheet.Cell(i + 2, 4).Value = string.Join(", ", data[i].RelatedDetails);  }  var stream = new MemoryStream();  workbook.SaveAs(stream);  stream.Position = 0;  return stream;  }  public async Task<List<DetailsReportViewModel>> GetDetailsByMachinesAndProductionsAsync(DateTime startDate,  DateTime endDate, CancellationToken ct)  {  return await context.Details  .Include(d => d.DetailProductions)!  .ThenInclude(dp => dp.Production)  .Include(d => d.DetailProducts)!  .ThenInclude(dp => dp.Product)  .ThenInclude(p => p!.Machine)  .Where(d => d.CreationDate >= startDate && d.CreationDate <= endDate)  .Select(d => new DetailsReportViewModel  {  DetailName = d.Name,  CreationDate = d.CreationDate,  Material = d.Material.ToString(),  RelatedProductions = d.DetailProductions!.Select(dp => dp.Production!.Name).ToList(),  RelatedMachines = d.DetailProducts!.Select(dp => dp.Product!.Machine!.Model).Distinct().ToList(),  QuantityInProducts = d.DetailProducts!.Sum(dp => dp.Quantity)  })  .ToListAsync(ct);  }  public async Task<Stream> CreatePdfDocumentDetailsByMachinesAndProductionsAsync(  List<DetailsReportViewModel> data,  DateTime startDate,  DateTime endDate,  CancellationToken ct)  {  logger.LogInformation("CreatePdfDocumentDetailsByMachinesAndProductionsAsync: {a}, {b}, {c}", data.Count, startDate.ToString(), endDate.ToString());  // HTML-шаблон для PDF  var htmlContent = new StringBuilder();  htmlContent.Append(@"  <html>  <head>  <meta charset='UTF-8'>  <style>  body { font-family: Arial, sans-serif; margin: 20px; }  h1 { text-align: center; font-size: 24px; }  table { width: 100%; border-collapse: collapse; margin-top: 20px; }  th, td { border: 1px solid black; padding: 8px; text-align: left; }  th { background-color: #f2f2f2; font-weight: bold; }  </style>  </head>  <body>  <h1>Отчет по деталям за период ");  htmlContent.Append($"{startDate:dd.MM.yyyy} - {endDate:dd.MM.yyyy}");  htmlContent.Append(@"</h1>  <table>  <tr>  <th>Деталь</th>  <th>Дата создания</th>  <th>Материал</th>  <th>Производства</th>  <th>Станки</th>  <th>Количество в продуктах</th>  </tr>");  // Добавление данных в таблицу  foreach (var item in data)  {  htmlContent.Append("<tr>");  htmlContent.Append($"<td>{System.Web.HttpUtility.HtmlEncode(item.DetailName)}</td>");  htmlContent.Append($"<td>{item.CreationDate:dd.MM.yyyy}</td>");  htmlContent.Append($"<td>{System.Web.HttpUtility.HtmlEncode(item.Material)}</td>");  htmlContent.Append(  $"<td>{System.Web.HttpUtility.HtmlEncode(string.Join(", ", item.RelatedProductions))}</td>");  htmlContent.Append(  $"<td>{System.Web.HttpUtility.HtmlEncode(string.Join(", ", item.RelatedMachines))}</td>");  htmlContent.Append($"<td>{item.QuantityInProducts}</td>");  htmlContent.Append("</tr>");  }  htmlContent.Append(@"  </table>  </body>  </html>");  // Временные файлы  string tempHtmlPath = Path.Combine(Path.GetTempPath(), $"{Guid.NewGuid()}.html");  string tempPdfPath = Path.Combine(Path.GetTempPath(), $"{Guid.NewGuid()}.pdf");  var stream = new MemoryStream();  try  {  // Сохраняем HTML во временный файл  await File.WriteAllTextAsync(tempHtmlPath, htmlContent.ToString(), Encoding.UTF8, ct);  // Запускаем wkhtmltopdf  var processInfo = new ProcessStartInfo  {  FileName = "wkhtmltopdf",  Arguments = $"--encoding UTF-8 {tempHtmlPath} {tempPdfPath}",  RedirectStandardOutput = true,  RedirectStandardError = true,  UseShellExecute = false,  CreateNoWindow = true  };  using (var process = new Process { StartInfo = processInfo })  {  process.Start();  await process.WaitForExitAsync(ct);  if (process.ExitCode != 0)  {  string error = await process.StandardError.ReadToEndAsync();  throw new Exception($"Ошибка wkhtmltopdf: {error}");  }  }  // Читаем PDF в поток  byte[] pdfBytes = await File.ReadAllBytesAsync(tempPdfPath, ct);  await stream.WriteAsync(pdfBytes, 0, pdfBytes.Length, ct);  stream.Position = 0;  return stream;  }  catch (Exception ex)  {  throw new Exception($"Ошибка при создании PDF: {ex.Message}", ex);  }  finally  {  // Очистка временных файлов  if (File.Exists(tempHtmlPath)) File.Delete(tempHtmlPath);  if (File.Exists(tempPdfPath)) File.Delete(tempPdfPath);  }  }  public async Task SendEmailAsync(Stream fileStream, string recipientEmail, string subject, string fileName,  string contentType)  {  var smtpClient = new SmtpClient("smtp.yandex.ru", 587)  {  Credentials = new NetworkCredential("vasmaae@yandex.ru", "yyrunxfhsdtprmss"),  EnableSsl = true    };  var message = new MailMessage("vasmaae@yandex.ru", recipientEmail)  {  Subject = subject,  Body = "Файл прикреплен.",  IsBodyHtml = false  };  fileStream.Position = 0;  message.Attachments.Add(new Attachment(fileStream, fileName, contentType));  await smtpClient.SendMailAsync(message);  }  }  // DetailBusinessLogicContract.cs  using System.Text.Json;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.StoragesContracts;  using Microsoft.Extensions.Logging;  namespace GoToWorkBusinessLogic.Implementations;  public class DetailBusinessLogicContract(  IDetailStorageContract detailStorageContract,  ILogger logger) : IDetailBusinessLogicContract  {  public List<DetailDataModel> GetAllDetails()  {  logger.LogInformation("Getting all details");  return detailStorageContract.GetList() ?? throw new NullListException();  }  public List<DetailDataModel> GetDetailsByCreationDate(DateTime? from = null, DateTime? to = null)  {  logger.LogInformation("Getting details by creation date from {from} to {to}", from, to);  if (from > to) throw new IncorrectDatesException((DateTime)from, (DateTime)to);  return detailStorageContract.GetList(from, to) ?? throw new NullListException();  }  public DetailDataModel GetDetailByData(string data)  {  logger.LogInformation("Getting detail by data: {data}", data);  if (data.IsEmpty()) throw new ArgumentNullException(nameof(data));  if (data.IsGuid())  return detailStorageContract.GetElementById(data) ?? throw new ElementNotFoundException(data);  return detailStorageContract.GetElementByName(data) ?? throw new ElementNotFoundException(data);  }  public void InsertDetail(DetailDataModel detail)  {  logger.LogInformation("Inserting new detail: {json}", JsonSerializer.Serialize(detail));  ArgumentNullException.ThrowIfNull(detail);  detail.Validate();  detailStorageContract.AddElement(detail);  }  public void UpdateDetail(DetailDataModel detail)  {  logger.LogInformation("Updating detail: {json}", JsonSerializer.Serialize(detail));  ArgumentNullException.ThrowIfNull(detail);  detail.Validate();  detailStorageContract.UpdElement(detail);  }  public void DeleteDetail(string id)  {  logger.LogInformation("Deleting detail with id: {id}", id);  if (string.IsNullOrWhiteSpace(id)) throw new ArgumentNullException(nameof(id));  if (!id.IsGuid()) throw new ValidationException("Id is not a unique identifier");  detailStorageContract.DelElement(id);  }  }  // ProductionBusinessLogicContract.cs  using System.Text.Json;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.StoragesContracts;  using Microsoft.Extensions.Logging;  namespace GoToWorkBusinessLogic.Implementations;  public class ProductionBusinessLogicContract(  IProductionStorageContract productionStorageContract,  ILogger logger) : IProductionBusinessLogicContract  {  public List<ProductionDataModel> GetAllProductions()  {  logger.LogInformation("Getting all productions");  return productionStorageContract.GetList() ?? throw new NullListException();  }    public ProductionDataModel GetProductionByData(string data)  {  logger.LogInformation("Getting production by data: {data}", data);  if (data.IsEmpty()) throw new ArgumentNullException(nameof(data));  if (data.IsGuid())  return productionStorageContract.GetElementById(data) ?? throw new ElementNotFoundException(data);  throw new ElementNotFoundException(data);  }  public void InsertProduction(ProductionDataModel production)  {  logger.LogInformation("Inserting new production: {json}", JsonSerializer.Serialize(production));  ArgumentNullException.ThrowIfNull(production);  production.Validate();  productionStorageContract.AddElement(production);  }  public void UpdateProduction(ProductionDataModel production)  {  logger.LogInformation("Updating production: {json}", JsonSerializer.Serialize(production));  ArgumentNullException.ThrowIfNull(production);  production.Validate();  productionStorageContract.UpdElement(production);  }  public void DeleteProduction(string id)  {  logger.LogInformation("Deleting production with id: {id}", id);  if (id.IsEmpty()) throw new ArgumentNullException(nameof(id));  if (!id.IsGuid()) throw new ValidationException("Id is not a unique identifier");  productionStorageContract.DelElement(id);  }  }  // MachineBusinessLogicContract.cs  using System.Text.Json;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.StoragesContracts;  using Microsoft.Extensions.Logging;  namespace GoToWorkBusinessLogic.Implementations;  public class MachineBusinessLogicContract(  IMachineStorageContract machineStorageContract,  ILogger logger) : IMachineBusinessLogicContract  {  public List<MachineDataModel> GetAllMachines()  {  logger.LogInformation("Getting all machines");  return machineStorageContract.GetList() ?? throw new NullListException();  }  public MachineDataModel GetMachineByData(string data)  {  logger.LogInformation("Getting machine by data: {data}", data);  if (data.IsEmpty()) throw new ArgumentNullException(nameof(data));  if (data.IsGuid())  return machineStorageContract.GetElementById(data) ?? throw new ElementNotFoundException(data);  return machineStorageContract.GetElementByModel(data) ?? throw new ElementNotFoundException(data);  }  public void InsertMachine(MachineDataModel machine)  {  logger.LogInformation("Inserting new machine: {json}", JsonSerializer.Serialize(machine));  ArgumentNullException.ThrowIfNull(machine);  machine.Validate();  machineStorageContract.AddElement(machine);  }  public void UpdateMachine(MachineDataModel machine)  {  logger.LogInformation("Updating machine: {json}", JsonSerializer.Serialize(machine));  ArgumentNullException.ThrowIfNull(machine);  machine.Validate();  machineStorageContract.UpdElement(machine);  }  public void DeleteMachine(string id)  {  logger.LogInformation("Deleting machine with id: {id}", id);  if (id.IsEmpty()) throw new ArgumentNullException(nameof(id));  if (!id.IsGuid()) throw new ValidationException("Id is not a unique identifier");  machineStorageContract.DelElement(id);  }  }  // EmployeeBusinessLogicContract.cs  using System.Text.Json;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.StoragesContracts;  using Microsoft.Extensions.Logging;  namespace GoToWorkBusinessLogic.Implementations;  public class EmployeeBusinessLogicContract(  IEmployeeStorageContract employeeStorageContract,  ILogger logger) : IEmployeeBusinessLogicContract  {  public List<EmployeeDataModel> GetAllEmployees()  {  logger.LogInformation("Getting all employees");  return employeeStorageContract.GetList()  ?? [];  }  public EmployeeDataModel GetEmployeeByData(string data)  {  logger.LogInformation("Getting employee by id: {data}", data);  if (data.IsEmpty()) throw new ArgumentNullException(nameof(data));  if (data.IsGuid())  return employeeStorageContract.GetElementById(data) ?? throw new ElementNotFoundException(data);  throw new ElementNotFoundException(data);  }  public void InsertEmployee(EmployeeDataModel employee)  {  logger.LogInformation("Inserting new employee: {json}", JsonSerializer.Serialize(employee));  ArgumentNullException.ThrowIfNull(employee);  employee.Validate();  employeeStorageContract.AddElement(employee);  }  public void UpdateEmployee(EmployeeDataModel employee)  {  logger.LogInformation("Updating employee: {json}", JsonSerializer.Serialize(employee));  ArgumentNullException.ThrowIfNull(employee);  employee.Validate();  employeeStorageContract.UpdElement(employee);  }  public void DeleteEmployee(string id)  {  logger.LogInformation("Deleting employee with id: {id}", id);  if (id.IsEmpty()) throw new ArgumentNullException(nameof(id));  if (!id.IsGuid()) throw new ValidationException("Id is not a unique identifier");  employeeStorageContract.DelElement(id);  }  }  // UserBusinessLogicContract.cs  using System.Text.Json;  using System.Text.RegularExpressions;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.StoragesContracts;  using Microsoft.Extensions.Logging;  using System.Security.Cryptography;  using System.Text;  using GoToWorkContracts.Enums;  namespace GoToWorkBusinessLogic.Implementations;  public class UserBusinessLogicContract(  IUserStorageContract userStorageContract,  ILogger logger) : IUserBusinessLogicContract  {  public List<UserDataModel> GetAllUsers()  {  logger.LogInformation("Getting all users");  return userStorageContract.GetList() ?? throw new NullListException();  }  public UserDataModel GetUserByData(string data)  {  logger.LogInformation("Getting user by data: {data}", data);  if (data.IsEmpty()) throw new ArgumentNullException(nameof(data));  if (data.IsGuid())  return userStorageContract.GetElementById(data) ?? throw new ElementNotFoundException(data);  if (RegexExtensions.EmailRegex().IsMatch(data))  return userStorageContract.GetElementByEmail(data) ?? throw new ElementNotFoundException(data);  return userStorageContract.GetElementByLogin(data) ?? throw new ElementNotFoundException(data);  }  public void InsertUser(UserDataModel user)  {  logger.LogInformation("Inserting new user: {json}", JsonSerializer.Serialize(user));  ArgumentNullException.ThrowIfNull(user);  user.Validate();  userStorageContract.AddElement(user);  }  public void UpdateUser(UserDataModel user)  {  logger.LogInformation("Updating user: {json}", JsonSerializer.Serialize(user));  ArgumentNullException.ThrowIfNull(user);  user.Validate();  userStorageContract.UpdElement(user);  }  public void DeleteUser(string id)  {  logger.LogInformation("Deleting user with id: {id}", id);  if (id.IsEmpty()) throw new ArgumentNullException(nameof(id));  if (!id.IsGuid()) throw new ValidationException("Id is not a unique identifier");  userStorageContract.DelElement(id);  }  public string Register(UserDataModel model)  {  logger.LogInformation("Registering new user: {json}", JsonSerializer.Serialize(model));  ArgumentNullException.ThrowIfNull(model);  var user = new UserDataModel(Guid.NewGuid().ToString(), model.Login, model.Email, model.Password, model.Role);  user.Validate();  user.Password = HashPassword(model.Password);  if (userStorageContract.GetElementByLogin(user.Login) != null ||  userStorageContract.GetElementByEmail(user.Email) != null)  {  throw new ElementExistsException(nameof(user.Login), user.Login);  }  userStorageContract.AddElement(user);  return user.Id;  }  public (string id, string login, UserRole role)? Login(string loginOrEmail, string password)  {  logger.LogInformation("User login attempt: {login}", loginOrEmail);  var user = RegexExtensions.EmailRegex().IsMatch(loginOrEmail)  ? userStorageContract.GetElementByEmail(loginOrEmail)  : userStorageContract.GetElementByLogin(loginOrEmail);  if (user == null)  {  logger.LogWarning("User not found: {login}", loginOrEmail);  return null;  }  if (user.Password != HashPassword(password))  {  logger.LogWarning("Invalid password for user: {login}", loginOrEmail);  return null;  }  return (user.Id, user.Login, user.Role);  }  private static string HashPassword(string password)  {  using (var sha256 = SHA256.Create())  {  var hashedBytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(password));  return BitConverter.ToString(hashedBytes).Replace("-", "").ToLower();  }  }  }  // WorkshopBusinessLogicContract.cs  using System.Text.Json;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.StoragesContracts;  using Microsoft.Extensions.Logging;  namespace GoToWorkBusinessLogic.Implementations;  public class WorkshopBusinessLogicContract(  IWorkshopStorageContract workshopStorageContract,  ILogger logger) : IWorkshopBusinessLogicContract  {  public List<WorkshopDataModel> GetAllWorkshops()  {  logger.LogInformation("Getting all workshops");  return workshopStorageContract.GetList() ?? throw new NullListException();  }  public List<WorkshopDataModel> GetWorkshopsByProduction(string productionId)  {  logger.LogInformation("Getting workshops by production ID: {productionId}", productionId);  if (productionId.IsEmpty()) throw new ArgumentNullException(nameof(productionId));  if (!productionId.IsGuid()) throw new ValidationException("ProductionId is not a unique identifier");  return workshopStorageContract.GetList(productionId) ?? throw new NullListException();  }  public WorkshopDataModel GetWorkshopByData(string data)  {  logger.LogInformation("Getting workshop by data: {data}", data);  if (data.IsEmpty()) throw new ArgumentNullException(nameof(data));  if (data.IsGuid())  return workshopStorageContract.GetElementById(data) ?? throw new ElementNotFoundException(data);  return workshopStorageContract.GetElementByAddress(data) ?? throw new ElementNotFoundException(data);  }  public void InsertWorkshop(WorkshopDataModel workshop)  {  logger.LogInformation("Inserting new workshop: {json}", JsonSerializer.Serialize(workshop));  ArgumentNullException.ThrowIfNull(workshop);  workshop.Validate();  workshopStorageContract.AddElement(workshop);  }  public void UpdateWorkshop(WorkshopDataModel workshop)  {  logger.LogInformation("Updating workshop: {json}", JsonSerializer.Serialize(workshop));  ArgumentNullException.ThrowIfNull(workshop);  workshop.Validate();  workshopStorageContract.UpdElement(workshop);  }  public void DeleteWorkshop(string id)  {  logger.LogInformation("Deleting workshop with Id: {id}", id);  if (id.IsEmpty()) throw new ArgumentNullException(nameof(id));  if (!id.IsGuid()) throw new ValidationException("Id is not a unique identifier");  workshopStorageContract.DelElement(id);  }  }  // GoToWorkBusinessLogic.csproj.nuget.g.targets  ﻿<?xml version="1.0" encoding="utf-8" standalone="no"?>  <Project ToolsVersion="14.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">  <ImportGroup Condition=" '$(ExcludeRestorePackageImports)' != 'true' ">  <Import Project="$(NuGetPackageRoot)microsoft.extensions.logging.abstractions/9.0.4/buildTransitive/net8.0/Microsoft.Extensions.Logging.Abstractions.targets" Condition="Exists('$(NuGetPackageRoot)microsoft.extensions.logging.abstractions/9.0.4/buildTransitive/net8.0/Microsoft.Extensions.Logging.Abstractions.targets')" />  <Import Project="$(NuGetPackageRoot)microsoft.extensions.options/9.0.4/buildTransitive/net8.0/Microsoft.Extensions.Options.targets" Condition="Exists('$(NuGetPackageRoot)microsoft.extensions.options/9.0.4/buildTransitive/net8.0/Microsoft.Extensions.Options.targets')" />  </ImportGroup>  </Project>  // GoToWorkBusinessLogic.GlobalUsings.g.cs  // <auto-generated/>  global using global::System;  global using global::System.Collections.Generic;  global using global::System.IO;  global using global::System.Linq;  global using global::System.Net.Http;  global using global::System.Threading;  global using global::System.Threading.Tasks;  // GoToWorkBusinessLogic.AssemblyInfo.cs  //------------------------------------------------------------------------------  // <auto-generated>  // This code was generated by a tool.  //  // Changes to this file may cause incorrect behavior and will be lost if  // the code is regenerated.  // </auto-generated>  //------------------------------------------------------------------------------  using System;  using System.Reflection;  [assembly: System.Reflection.AssemblyCompanyAttribute("GoToWorkBusinessLogic")]  [assembly: System.Reflection.AssemblyConfigurationAttribute("Release")]  [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]  [assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+8753cd23468d9b41857b154ffbb5889134ac4e4c")]  [assembly: System.Reflection.AssemblyProductAttribute("GoToWorkBusinessLogic")]  [assembly: System.Reflection.AssemblyTitleAttribute("GoToWorkBusinessLogic")]  [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("GoToWorkApp")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("DynamicProxyGenAssembly2")]  // Generated by the MSBuild WriteCodeFragment class.  // .NETCoreApp,Version=v9.0.AssemblyAttributes.cs  // <autogenerated />  using System;  using System.Reflection;  [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v9.0", FrameworkDisplayName = ".NET 9.0")]  // GoToWorkBusinessLogic.GlobalUsings.g.cs  // <auto-generated/>  global using global::System;  global using global::System.Collections.Generic;  global using global::System.IO;  global using global::System.Linq;  global using global::System.Net.Http;  global using global::System.Threading;  global using global::System.Threading.Tasks;  // GoToWorkBusinessLogic.AssemblyInfo.cs  //------------------------------------------------------------------------------  // <auto-generated>  // This code was generated by a tool.  //  // Changes to this file may cause incorrect behavior and will be lost if  // the code is regenerated.  // </auto-generated>  //------------------------------------------------------------------------------  using System;  using System.Reflection;  [assembly: System.Reflection.AssemblyCompanyAttribute("GoToWorkBusinessLogic")]  [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]  [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]  [assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+8753cd23468d9b41857b154ffbb5889134ac4e4c")]  [assembly: System.Reflection.AssemblyProductAttribute("GoToWorkBusinessLogic")]  [assembly: System.Reflection.AssemblyTitleAttribute("GoToWorkBusinessLogic")]  [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("GoToWorkApp")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("DynamicProxyGenAssembly2")]  // Generated by the MSBuild WriteCodeFragment class.  // .NETCoreApp,Version=v9.0.AssemblyAttributes.cs  // <autogenerated />  using System;  using System.Reflection;  [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v9.0", FrameworkDisplayName = ".NET 9.0")]  // IEmployeeBusinessLogicContract.cs  using GoToWorkContracts.DataModels;  namespace GoToWorkContracts.BusinessLogicContracts;  public interface IEmployeeBusinessLogicContract  {  List<EmployeeDataModel> GetAllEmployees();  EmployeeDataModel GetEmployeeByData(string data);  void InsertEmployee(EmployeeDataModel employee);  void UpdateEmployee(EmployeeDataModel employee);  void DeleteEmployee(string id);  }  // IReportContract.cs  using GoToWorkContracts.DataModels;  using GoToWorkContracts.ViewModels;  namespace GoToWorkContracts.BusinessLogicContracts;  public interface IReportContract  {  Task<List<WorkshopsReportViewModel>> GetWorkshopsByDetailsAsync(List<string> selectedDetailIds,  CancellationToken ct);  Task<Stream> CreateDocxDocumentWorkshopsByDetailsAsync(List<WorkshopsReportViewModel> data, CancellationToken ct);  Task<Stream> CreateXlsxDocumentWorkshopsByDetailsAsync(List<WorkshopsReportViewModel> data, CancellationToken ct);  Task<List<DetailsReportViewModel>> GetDetailsByMachinesAndProductionsAsync(  DateTime startDate, DateTime endDate, CancellationToken ct);  Task<Stream> CreatePdfDocumentDetailsByMachinesAndProductionsAsync(List<DetailsReportViewModel> data,  DateTime startDate, DateTime endDate, CancellationToken ct);    public Task SendEmailAsync(Stream fileStream, string recipientEmail, string subject, string fileName,  string contentType);  }  // IMachineBusinessLogicContract.cs  using GoToWorkContracts.DataModels;  namespace GoToWorkContracts.BusinessLogicContracts;  public interface IMachineBusinessLogicContract  {  List<MachineDataModel> GetAllMachines();  MachineDataModel GetMachineByData(string data);  void InsertMachine(MachineDataModel machine);  void UpdateMachine(MachineDataModel machine);  void DeleteMachine(string id);  }  // IDetailBusinessLogicContract.cs  using GoToWorkContracts.DataModels;  namespace GoToWorkContracts.BusinessLogicContracts;  public interface IDetailBusinessLogicContract  {  List<DetailDataModel> GetAllDetails();  List<DetailDataModel> GetDetailsByCreationDate(DateTime? from = null, DateTime? to = null);  DetailDataModel GetDetailByData(string data);  void InsertDetail(DetailDataModel detail);  void UpdateDetail(DetailDataModel detail);  void DeleteDetail(string id);  }  // IProductionBusinessLogicContract.cs  using GoToWorkContracts.DataModels;  namespace GoToWorkContracts.BusinessLogicContracts;  public interface IProductionBusinessLogicContract  {  List<ProductionDataModel> GetAllProductions();  ProductionDataModel GetProductionByData(string data);  void InsertProduction(ProductionDataModel production);  void UpdateProduction(ProductionDataModel production);  void DeleteProduction(string id);  }  // IWorkshopBusinessLogicContract.cs  using GoToWorkContracts.DataModels;  namespace GoToWorkContracts.BusinessLogicContracts;  public interface IWorkshopBusinessLogicContract  {  List<WorkshopDataModel> GetAllWorkshops();  List<WorkshopDataModel> GetWorkshopsByProduction(string productionId);  WorkshopDataModel GetWorkshopByData(string data);  void InsertWorkshop(WorkshopDataModel workshop);  void UpdateWorkshop(WorkshopDataModel workshop);  void DeleteWorkshop(string id);  }  // IProductBusinessLogicContract.cs  using GoToWorkContracts.DataModels;  namespace GoToWorkContracts.BusinessLogicContracts;  public interface IProductBusinessLogicContract  {  List<ProductDataModel> GetAllProducts();  List<ProductDataModel> GetProductsByMachine(string machineId);  List<ProductDataModel> GetProductsByCreationDate(DateTime from, DateTime to);  ProductDataModel GetProductByData(string data);  void InsertProduct(ProductDataModel product);  void UpdateProduct(ProductDataModel product);  void DeleteProduct(string id);  }  // IUserBusinessLogicContract.cs  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Enums;  namespace GoToWorkContracts.BusinessLogicContracts;  public interface IUserBusinessLogicContract  {  List<UserDataModel> GetAllUsers();  UserDataModel GetUserByData(string data);  void InsertUser(UserDataModel user);  void UpdateUser(UserDataModel user);  void DeleteUser(string id);  string Register(UserDataModel model);  (string id, string login, UserRole role)? Login(string loginOrEmail, string password);  }  // IEmailContract.cs  namespace GoToWorkContracts.BusinessLogicContracts;  public interface IEmailContract  {  public Task SendEmailAsync(string to, string subject, string body, string? attachmentPath = null);  }  // MaterialType.cs  namespace GoToWorkContracts.Enums;  public enum MaterialType  {  None = 0,  Steel = 1,  Aluminum = 2,  Plastic = 3,  Copper = 4,  Composite = 5  }  // MachineType.cs  namespace GoToWorkContracts.Enums;  public enum MachineType  {  None = 0,  Lathe = 1,  MillingMachine = 2,  DrillPress = 3,  BandSaw = 4,  CNC = 5  }  // UserRole.cs  namespace GoToWorkContracts.Enums;  public enum UserRole  {  None = 0,  Executor = 1,  Guarantor = 2  }  // StringExtensions.cs  namespace GoToWorkContracts.Extensions;  public static class StringExtensions  {  public static bool IsEmpty(this string str) => string.IsNullOrWhiteSpace(str);  public static bool IsGuid(this string str) => Guid.TryParse(str, out \_);  }  // RegexExtensions.cs  using System.Text.RegularExpressions;  namespace GoToWorkContracts.Extensions;  public static partial class RegexExtensions  {  [GeneratedRegex(@"^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$")]  public static partial Regex EmailRegex();    [GeneratedRegex(@"^(?=.\*[A-Z])(?=.\*\W).{8,}$")]  public static partial Regex PasswordRegex();  }  // EmailBindingModel.cs  namespace GoToWorkContracts.BindingModels;  public class EmailBindingModel  {  public string To { get; set; } = string.Empty;  public string Subject { get; set; } = string.Empty;  public string Body { get; set; } = string.Empty;  public string? AttachmentPath { get; set; }  }  // DetailsReportBindingModel.cs  namespace GoToWorkContracts.BindingModels;  public class DetailsReportBindingModel  {  public DateTime startDate { get; set; }  public DateTime endDate { get; set; }  public string? email { get; set; }  }  // UserRegisterBindingModel.cs  using GoToWorkContracts.Enums;  namespace GoToWorkContracts.BindingModels;  public class UserRegisterBindingModel  {  public string? Login { get; set; }  public string? Email { get; set; }  public string? Password { get; set; }  public UserRole Role { get; set; }  }  // DetailProductionBindingModel.cs  namespace GoToWorkContracts.BindingModels;  public class DetailProductionBindingModel  {  public string? DetailId { get; set; }  public string? ProductionId { get; set; }  }  // EmployeeWorkshopBindingModel.cs  namespace GoToWorkContracts.BindingModels;  public class EmployeeWorkshopBindingModel  {  public string? EmployeeId { get; set; }  public string? WorkshopId { get; set; }  }  // ProductionBindingModel.cs  namespace GoToWorkContracts.BindingModels;  public class ProductionBindingModel  {  public string? Id { get; set; }  public string? Name { get; set; }  public List<DetailProductionBindingModel>? Details { get; set; }  public List<WorkshopBindingModel>? Workshops { get; set; }  }  // UserBindingModel.cs  using GoToWorkContracts.Enums;  namespace GoToWorkContracts.BindingModels;  public class UserBindingModel  {  public string? Id { get; set; }  public string? Login { get; set; }  public string? Email { get; set; }  public string? Password { get; set; }  public UserRole Role { get; set; }  }  // ProductBindingModel.cs  namespace GoToWorkContracts.BindingModels;  public class ProductBindingModel  {  public string? Id { get; set; }  public string? MachineId { get; set; }  public string? Name { get; set; }  public DateTime CreationDate { get; set; }  public List<DetailProductBindingModel>? Details { get; set; }  }  // DetailProductBindingModel.cs  namespace GoToWorkContracts.BindingModels;  public class DetailProductBindingModel  {  public string? DetailId { set; get; }  public string? ProductId { get; set; }  public int Quantity { get; set; }  }  // EmployeeBindingModel.cs  namespace GoToWorkContracts.BindingModels;  public class EmployeeBindingModel  {  public string? Id { get; set; }  public string? FullName { get; set; }  }  // WorkshopsReportBindingModel.cs  namespace GoToWorkContracts.BindingModels;  public class WorkshopsReportBindingModel  {  public List<string> DetailIds { get; set; }  }  // DetailBindingModel.cs  using GoToWorkContracts.Enums;  namespace GoToWorkContracts.BindingModels;  public class DetailBindingModel  {  public string? Id { get; set; }  public string? Name { get; set; }  public MaterialType Material { get; set; }  public DateTime CreationDate { get; set; }  }  // UserLoginBindingModel.cs  namespace GoToWorkContracts.BindingModels;  public class UserLoginBindingModel  {  public string Login { get; set; } = string.Empty;  public string Password { get; set; } = string.Empty;  }  // WorkshopBindingModel.cs  namespace GoToWorkContracts.BindingModels;  public class WorkshopBindingModel  {  public string? Id { get; set; }  public string? ProductionId { get; set; }  public string? Address { get; set; }  public List<EmployeeWorkshopBindingModel>? Employees { get; set; }  }  // MachineBindingModel.cs  using GoToWorkContracts.Enums;  namespace GoToWorkContracts.BindingModels;  public class MachineBindingModel  {  public string? Id { get; set; }  public string? Model { get; set; }  public MachineType Type { get; set; }  public List<EmployeeMachineBindingModel>? Employees { get; set; }  public List<ProductBindingModel>? Products { get; set; }  }  // EmployeeMachineBindingModel.cs  namespace GoToWorkContracts.BindingModels;  public class EmployeeMachineBindingModel  {  public string? EmployeeId { get; set; }  public string? MachineId { get; set; }  }  // ProductionViewModel.cs  using System.Collections.Generic;  namespace GoToWorkContracts.ViewModels;  public class ProductionViewModel  {  public required string Id { get; set; }  public required string Name { get; set; }  public required List<DetailProductionViewModel>? Details { get; set; }  public required List<WorkshopViewModel>? Workshops { get; set; }  }  // EmployeeWorkshopViewModel.cs  namespace GoToWorkContracts.ViewModels;  public class EmployeeWorkshopViewModel  {  public required string EmployeeId { get; set; }  public required string WorkshopId { get; set; }  public required string EmployeeName { get; set; }  public required string WorkshopName { get; set; }  }  // DetailsReportViewModel.cs  namespace GoToWorkContracts.ViewModels;  public class DetailsReportViewModel  {  public string DetailName { get; set; }  public DateTime CreationDate { get; set; }  public string Material { get; set; }  public List<string> RelatedProductions { get; set; }  public List<string> RelatedMachines { get; set; }  public int QuantityInProducts { get; set; }  }  // ProductViewModel.cs  using System;  using System.Collections.Generic;  namespace GoToWorkContracts.ViewModels;  public class ProductViewModel  {  public required string Id { get; set; }  public required string? MachineId { get; set; }  public required string Name { get; set; }  public DateTime CreationDate { get; set; }  public required string? MachineName { get; set; }  public required List<DetailProductViewModel>? Details { get; set; }  }  // WorkshopsReportViewModel.cs  namespace GoToWorkContracts.ViewModels;  public class WorkshopsReportViewModel  {  public string WorkshopId { get; set; }  public string Address { get; set; }  public string ProductionName { get; set; }  public List<string> RelatedDetails { get; set; }  }  // EmployeeViewModel.cs  namespace GoToWorkContracts.ViewModels;  public class EmployeeViewModel  {  public required string Id { get; set; }  public required string FullName { get; set; }  }  // DetailProductionViewModel.cs  namespace GoToWorkContracts.ViewModels;  public class DetailProductionViewModel  {  public required string DetailId { get; set; }  public required string ProductionId { get; set; }  public required string DetailName { get; set; }  public required string ProductionName { get; set; }  }  // WorkshopViewModel.cs  using System.Collections.Generic;  namespace GoToWorkContracts.ViewModels;  public class WorkshopViewModel  {  public required string Id { get; set; }  public required string? ProductionId { get; set; }  public required string Address { get; set; }  public required string? ProductionName { get; set; }  public required List<EmployeeWorkshopViewModel>? Employees { get; set; }  }  // DetailProductViewModel.cs  namespace GoToWorkContracts.ViewModels;  public class DetailProductViewModel  {  public required string ProductId { get; set; }  public required string DetailId { get; set; }  public required string DetailName { get; set; }  public int Quantity { get; set; }  }  // EmployeeMachineViewModel.cs  namespace GoToWorkContracts.ViewModels;  public class EmployeeMachineViewModel  {  public required string EmployeeId { get; set; }  public required string MachineId { get; set; }  public required string EmployeeName { get; set; }  public required string MachineName { get; set; }  }  // UserViewModel.cs  using GoToWorkContracts.Enums;  namespace GoToWorkContracts.ViewModels;  public class UserViewModel  {  public required string Id { get; set; }  public required string Login { get; set; }  public required string Email { get; set; }  public required string Password { get; set; }  public UserRole Role { get; set; }  }  // TokenViewModel.cs  namespace GoToWorkContracts.ViewModels  {  public class TokenViewModel  {  public required string Token { get; set; }  }  }  // MachineViewModel.cs  using GoToWorkContracts.Enums;  using System.Collections.Generic;  namespace GoToWorkContracts.ViewModels;  public class MachineViewModel  {  public required string Id { get; set; }  public required string Model { get; set; }  public MachineType Type { get; set; }  public required List<EmployeeMachineViewModel>? Employees { get; set; }  public required List<ProductViewModel>? Products { get; set; }  }  // DetailViewModel.cs  using GoToWorkContracts.Enums;  namespace GoToWorkContracts.ViewModels;  public class DetailViewModel  {  public required string Id { get; set; }  public required string Name { get; set; }  public MaterialType Material { get; set; }  public DateTime CreationDate { get; set; }  }  // ValidationException.cs  namespace GoToWorkContracts.Exceptions;  public class ValidationException(string message) : Exception(message)  {  }  // StorageException.cs  ﻿namespace GoToWorkContracts.Exceptions;  public class StorageException(Exception ex)  : Exception($"Error while working in storage: {ex.Message}", ex);  // IncorrectDatesException.cs  ﻿namespace GoToWorkContracts.Exceptions;  public class IncorrectDatesException(DateTime start, DateTime end)  : Exception($"The end date must be later than the start date. " +  $"StartDate: {start:dd.MM.YYYY}. EndDate: {end:dd.MM.YYYY}");  // ElementDeletedException.cs  ﻿namespace GoToWorkContracts.Exceptions;  public class ElementDeletedException(string id)  : Exception($"Cannot modify a deleted item (id: {id})");  // ElementNotFoundException.cs  ﻿namespace GoToWorkContracts.Exceptions;  public class ElementNotFoundException(string value)  : Exception($"Element not found at value = {value}")  {  public string Value { get; private set; } = value;  }  // ElementExistsException.cs  ﻿namespace GoToWorkContracts.Exceptions;  public class ElementExistsException(string paramName, string paramValue)  : Exception($"There is already an element with value {paramValue} of parameter {paramName}")  {  public string ParamName { get; private set; } = paramName;  public string ParamValue { get; private set; } = paramValue;  }  // NullListException.cs  ﻿namespace GoToWorkContracts.Exceptions;  public class NullListException()  : Exception("The returned list is null");  // IProductAdapter.cs  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  namespace GoToWorkContracts.AdapterContracts;  public interface IProductAdapter  {  ProductOperationResponse GetList();  ProductOperationResponse GetListByMachine(string machineId);  ProductOperationResponse GetListByCreationDate(DateTime from, DateTime to);  ProductOperationResponse GetElement(string data);  ProductOperationResponse CreateProduct(ProductBindingModel productModel);  ProductOperationResponse UpdateProduct(ProductBindingModel productModel);  ProductOperationResponse DeleteProduct(string id);  }  // IReportAdapter.cs  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  using GoToWorkContracts.ViewModels;  namespace GoToWorkContracts.AdapterContracts;  public interface IReportAdapter  {  Task<ReportOperationResponse> GetWorkshopsByDetailsAsync(  WorkshopsReportBindingModel selectedDetailIds, CancellationToken ct);  Task<ReportOperationResponse> GetDetailsByMachinesAndProductionsAsync(  DetailsReportBindingModel selectedDates, CancellationToken ct);  Task<ReportOperationResponse> CreateDocxDocumentWorkshopsByDetailsAsync(  WorkshopsReportBindingModel selectedDetailIds, CancellationToken ct);  Task<ReportOperationResponse> CreateXlsxDocumentWorkshopsByDetailsAsync(  WorkshopsReportBindingModel selectedDetailIds, CancellationToken ct);  Task<ReportOperationResponse> CreatePdfDocumentDetailsByMachinesAndProductionsAsync(  DetailsReportBindingModel selectedDates, CancellationToken ct);  Task<ReportOperationResponse> SendPdfDocumentDetailsByMachinesAndProductionsEmailAsync(  DetailsReportBindingModel selectedDates, CancellationToken ct);  }  // IProductionAdapter.cs  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  namespace GoToWorkContracts.AdapterContracts;  public interface IProductionAdapter  {  ProductionOperationResponse GetList();  ProductionOperationResponse GetElement(string data);  ProductionOperationResponse CreateProduction(ProductionBindingModel productionModel);  ProductionOperationResponse UpdateProduction(ProductionBindingModel productionModel);  ProductionOperationResponse DeleteProduction(string id);  }  // IUserAdapter.cs  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  namespace GoToWorkContracts.AdapterContracts;  public interface IUserAdapter  {  UserOperationResponse GetList();  UserOperationResponse GetElement(string data);  UserOperationResponse CreateUser(UserBindingModel userModel);  UserOperationResponse UpdateUser(UserBindingModel userModel);  UserOperationResponse DeleteUser(string id);  UserOperationResponse Register(UserRegisterBindingModel model);  AuthOperationResponse Login(UserLoginBindingModel model);  }  // IWorkshopAdapter.cs  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  namespace GoToWorkContracts.AdapterContracts;  public interface IWorkshopAdapter  {  WorkshopOperationResponse GetList();  WorkshopOperationResponse GetListByProduction(string productionId);  WorkshopOperationResponse GetElement(string data);  WorkshopOperationResponse CreateWorkshop(WorkshopBindingModel workshopModel);  WorkshopOperationResponse UpdateWorkshop(WorkshopBindingModel workshopModel);  WorkshopOperationResponse DeleteWorkshop(string id);  }  // IEmployeeAdapter.cs  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  namespace GoToWorkContracts.AdapterContracts;  public interface IEmployeeAdapter  {  EmployeeOperationResponse GetList();  EmployeeOperationResponse GetElement(string data);  EmployeeOperationResponse CreateEmployee(EmployeeBindingModel employeeModel);  EmployeeOperationResponse UpdateEmployee(EmployeeBindingModel employeeModel);  EmployeeOperationResponse DeleteEmployee(string id);  }  // IMachineAdapter.cs  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  namespace GoToWorkContracts.AdapterContracts;  public interface IMachineAdapter  {  MachineOperationResponse GetList();  MachineOperationResponse GetElement(string data);  MachineOperationResponse CreateMachine(MachineBindingModel machineModel);  MachineOperationResponse UpdateMachine(MachineBindingModel machineModel);  MachineOperationResponse DeleteMachine(string id);  }  // IDetailAdapter.cs  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  namespace GoToWorkContracts.AdapterContracts;  public interface IDetailAdapter  {  DetailOperationResponse GetList();  DetailOperationResponse GetElementsByCreationDate(DateTime? from = null, DateTime? to = null);  DetailOperationResponse GetElement(string data);  DetailOperationResponse CreateDetail(DetailBindingModel detailModel);  DetailOperationResponse UpdateDetail(DetailBindingModel detailModel);  DetailOperationResponse DeleteDetail(string id);  }  // UserOperationResponse.cs  using GoToWorkContracts.Infrastructure;  using GoToWorkContracts.ViewModels;  namespace GoToWorkContracts.AdapterContracts.OperationResponses;  public class UserOperationResponse : OperationResponse  {  public static UserOperationResponse OK(List<UserViewModel> data) =>  OK<UserOperationResponse, List<UserViewModel>>(data);  public static UserOperationResponse OK(UserViewModel data) =>  OK<UserOperationResponse, UserViewModel>(data);  public static UserOperationResponse NoContent() =>  NoContent<UserOperationResponse>();  public static UserOperationResponse NotFound(string message) =>  NotFound<UserOperationResponse>(message);  public static UserOperationResponse BadRequest(string message) =>  BadRequest<UserOperationResponse>(message);  public static UserOperationResponse InternalServerError(string message) =>  InternalServerError<UserOperationResponse>(message);  public static UserOperationResponse Unauthorized(string message) =>  Unauthorized<UserOperationResponse>(message);  }  // WorkshopOperationResponse.cs  using GoToWorkContracts.Infrastructure;  using GoToWorkContracts.ViewModels;  namespace GoToWorkContracts.AdapterContracts.OperationResponses;  public class WorkshopOperationResponse : OperationResponse  {  public static WorkshopOperationResponse OK(List<WorkshopViewModel> data) =>  OK<WorkshopOperationResponse, List<WorkshopViewModel>>(data);  public static WorkshopOperationResponse OK(WorkshopViewModel data) =>  OK<WorkshopOperationResponse, WorkshopViewModel>(data);  public static WorkshopOperationResponse NoContent() =>  NoContent<WorkshopOperationResponse>();  public static WorkshopOperationResponse NotFound(string message) =>  NotFound<WorkshopOperationResponse>(message);  public static WorkshopOperationResponse BadRequest(string message) =>  BadRequest<WorkshopOperationResponse>(message);  public static WorkshopOperationResponse InternalServerError(string message) =>  InternalServerError<WorkshopOperationResponse>(message);  public static WorkshopOperationResponse Unauthorized(string message) =>  Unauthorized<WorkshopOperationResponse>(message);  }  // DetailOperationResponse.cs  using GoToWorkContracts.Infrastructure;  using GoToWorkContracts.ViewModels;  namespace GoToWorkContracts.AdapterContracts.OperationResponses;  public class DetailOperationResponse : OperationResponse  {  public static DetailOperationResponse OK(List<DetailViewModel> data) =>  OK<DetailOperationResponse, List<DetailViewModel>>(data);  public static DetailOperationResponse OK(DetailViewModel data) =>  OK<DetailOperationResponse, DetailViewModel>(data);  public static DetailOperationResponse NoContent() =>  NoContent<DetailOperationResponse>();  public static DetailOperationResponse NotFound(string message) =>  NotFound<DetailOperationResponse>(message);  public static DetailOperationResponse BadRequest(string message) =>  BadRequest<DetailOperationResponse>(message);  public static DetailOperationResponse InternalServerError(string message) =>  InternalServerError<DetailOperationResponse>(message);  public static DetailOperationResponse Unauthorized(string message) =>  Unauthorized<DetailOperationResponse>(message);  }  // EmployeeOperationResponse.cs  using GoToWorkContracts.Infrastructure;  using GoToWorkContracts.ViewModels;  namespace GoToWorkContracts.AdapterContracts.OperationResponses;  public class EmployeeOperationResponse : OperationResponse  {  public static EmployeeOperationResponse OK(List<EmployeeViewModel> data) =>  OK<EmployeeOperationResponse, List<EmployeeViewModel>>(data);  public static EmployeeOperationResponse OK(EmployeeViewModel data) =>  OK<EmployeeOperationResponse, EmployeeViewModel>(data);  public static EmployeeOperationResponse NoContent() =>  NoContent<EmployeeOperationResponse>();  public static EmployeeOperationResponse NotFound(string message) =>  NotFound<EmployeeOperationResponse>(message);  public static EmployeeOperationResponse BadRequest(string message) =>  BadRequest<EmployeeOperationResponse>(message);  public static EmployeeOperationResponse InternalServerError(string message) =>  InternalServerError<EmployeeOperationResponse>(message);  public static EmployeeOperationResponse Unauthorized(string message) =>  Unauthorized<EmployeeOperationResponse>(message);  }  // ProductOperationResponse.cs  using GoToWorkContracts.Infrastructure;  using GoToWorkContracts.ViewModels;  namespace GoToWorkContracts.AdapterContracts.OperationResponses;  public class ProductOperationResponse : OperationResponse  {  public static ProductOperationResponse OK(List<ProductViewModel> data) =>  OK<ProductOperationResponse, List<ProductViewModel>>(data);  public static ProductOperationResponse OK(ProductViewModel data) =>  OK<ProductOperationResponse, ProductViewModel>(data);  public static ProductOperationResponse NoContent() =>  NoContent<ProductOperationResponse>();  public static ProductOperationResponse NotFound(string message) =>  NotFound<ProductOperationResponse>(message);  public static ProductOperationResponse BadRequest(string message) =>  BadRequest<ProductOperationResponse>(message);  public static ProductOperationResponse InternalServerError(string message) =>  InternalServerError<ProductOperationResponse>(message);  public static ProductOperationResponse Unauthorized(string message) =>  Unauthorized<ProductOperationResponse>(message);  }  // MachineOperationResponse.cs  using GoToWorkContracts.Infrastructure;  using GoToWorkContracts.ViewModels;  namespace GoToWorkContracts.AdapterContracts.OperationResponses;  public class MachineOperationResponse : OperationResponse  {  public static MachineOperationResponse OK(List<MachineViewModel> data) =>  OK<MachineOperationResponse, List<MachineViewModel>>(data);  public static MachineOperationResponse OK(MachineViewModel data) =>  OK<MachineOperationResponse, MachineViewModel>(data);  public static MachineOperationResponse NoContent() =>  NoContent<MachineOperationResponse>();  public static MachineOperationResponse NotFound(string message) =>  NotFound<MachineOperationResponse>(message);  public static MachineOperationResponse BadRequest(string message) =>  BadRequest<MachineOperationResponse>(message);  public static MachineOperationResponse InternalServerError(string message) =>  InternalServerError<MachineOperationResponse>(message);  public static MachineOperationResponse Unauthorized(string message) =>  Unauthorized<MachineOperationResponse>(message);  }  // AuthOperationResponse.cs  using GoToWorkContracts.Infrastructure;  using GoToWorkContracts.ViewModels;  namespace GoToWorkContracts.AdapterContracts.OperationResponses;  public class AuthOperationResponse : OperationResponse  {  public static AuthOperationResponse OK(TokenViewModel data) =>  OK<AuthOperationResponse, TokenViewModel>(data);  public static AuthOperationResponse BadRequest(string message) =>  BadRequest<AuthOperationResponse>(message);  public static AuthOperationResponse Unauthorized(string message) =>  Unauthorized<AuthOperationResponse>(message);  }  // ReportOperationResponse.cs  using GoToWorkContracts.Infrastructure;  using GoToWorkContracts.ViewModels;  namespace GoToWorkContracts.AdapterContracts.OperationResponses;  public class ReportOperationResponse : OperationResponse  {  public static ReportOperationResponse OK(List<WorkshopsReportViewModel> data) =>  OK<ReportOperationResponse, List<WorkshopsReportViewModel>>(data);  public static ReportOperationResponse OK(List<DetailsReportViewModel> data) =>  OK<ReportOperationResponse, List<DetailsReportViewModel>>(data);  public static ReportOperationResponse OK(Stream data, string filename) =>  OK<ReportOperationResponse, Stream>(data, filename);  public static ReportOperationResponse NoContent() =>  NoContent<ReportOperationResponse>();  public static ReportOperationResponse NotFound(string message) =>  NotFound<ReportOperationResponse>(message);  public static ReportOperationResponse BadRequest(string message) =>  BadRequest<ReportOperationResponse>(message);  public static ReportOperationResponse InternalServerError(string message) =>  InternalServerError<ReportOperationResponse>(message);  public static ReportOperationResponse Unauthorized(string message) =>  Unauthorized<ReportOperationResponse>(message);  }  // ProductionOperationResponse.cs  using GoToWorkContracts.Infrastructure;  using GoToWorkContracts.ViewModels;  namespace GoToWorkContracts.AdapterContracts.OperationResponses;  public class ProductionOperationResponse : OperationResponse  {  public static ProductionOperationResponse OK(List<ProductionViewModel> data) =>  OK<ProductionOperationResponse, List<ProductionViewModel>>(data);  public static ProductionOperationResponse OK(ProductionViewModel data) =>  OK<ProductionOperationResponse, ProductionViewModel>(data);  public static ProductionOperationResponse NoContent() =>  NoContent<ProductionOperationResponse>();  public static ProductionOperationResponse NotFound(string message) =>  NotFound<ProductionOperationResponse>(message);  public static ProductionOperationResponse BadRequest(string message) =>  BadRequest<ProductionOperationResponse>(message);  public static ProductionOperationResponse InternalServerError(string message) =>  InternalServerError<ProductionOperationResponse>(message);  public static ProductionOperationResponse Unauthorized(string message) =>  Unauthorized<ProductionOperationResponse>(message);  }  // UserDataModel.cs  using GoToWorkContracts.Enums;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.Infrastructure;  namespace GoToWorkContracts.DataModels;  public class UserDataModel : IValidation  {  public string Id { get; set; }  public string Login { get; set; }  public string Email { get; set; }  public string Password { get; set; }  public UserRole Role { get; set; }  public UserDataModel(string id, string login, string email, string password, UserRole role)  {  Id = id;  Login = login;  Email = email;  Password = password;  Role = role;  }  public UserDataModel()  {  }  public void Validate()  {  if (Id.IsEmpty())  throw new ValidationException("Field Id is empty");  if (!Id.IsGuid())  throw new ValidationException("The value in the field Id is not a Guid");  if (Login.IsEmpty())  throw new ValidationException("Field Login is empty");  if (Email.IsEmpty())  throw new ValidationException("Field Email is empty");  if (!RegexExtensions.EmailRegex().IsMatch(Email))  throw new ValidationException("Field Email is not a valid email address");  if (Password.IsEmpty())  throw new ValidationException("Field Password is empty");  if (!RegexExtensions.PasswordRegex().IsMatch(Password))  throw new ValidationException("Field Password is not a valid password");  if (Role == UserRole.None)  throw new ValidationException("Field Role is empty");  }  }  // EmployeeWorkshopDataModel.cs  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.Infrastructure;  namespace GoToWorkContracts.DataModels;  public class EmployeeWorkshopDataModel : IValidation  {  public string EmployeeId { get; set; }  public string WorkshopId { get; set; }  public string EmployeeName => \_employee?.FullName ?? string.Empty;  public string WorkshopName => \_workshop?.Address ?? string.Empty;  private readonly EmployeeDataModel? \_employee;  private readonly WorkshopDataModel? \_workshop;  public EmployeeWorkshopDataModel(string employeeId, string workshopId)  {  EmployeeId = employeeId;  WorkshopId = workshopId;  }  public EmployeeWorkshopDataModel(string employeeId, string workshopId, EmployeeDataModel? employee,  WorkshopDataModel? workshop) : this(employeeId, workshopId)  {  \_employee = employee;  \_workshop = workshop;  }  public EmployeeWorkshopDataModel()  {  }  public void Validate()  {  if (EmployeeId.IsEmpty())  throw new ValidationException("Field EmployeeId is empty");  if (!EmployeeId.IsGuid())  throw new ValidationException("The value in the field EmployeeId is not a Guid");  if (WorkshopId.IsEmpty())  throw new ValidationException("Field WorkshopId is empty");  if (!WorkshopId.IsGuid())  throw new ValidationException("The value in the field WorkshopId is not a Guid");  }  }  // DetailProductDataModel.cs  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.Infrastructure;  namespace GoToWorkContracts.DataModels;  public class DetailProductDataModel : IValidation  {  public string ProductId { get; set; }  public string DetailId { get; set; }  public int Quantity { get; set; }  public string DetailName => \_detail?.Name ?? string.Empty;  public string ProductName => \_product?.Name ?? string.Empty;  private readonly DetailDataModel? \_detail;  private readonly ProductDataModel? \_product;  public DetailProductDataModel(string productId, string detailId, int quantity)  {  ProductId = productId;  DetailId = detailId;  Quantity = quantity;  }  public DetailProductDataModel(string productId, string detailId, int quantity, DetailDataModel? detail,  ProductDataModel? product) : this(productId, detailId, quantity)  {  \_detail = detail;  \_product = product;  }  public DetailProductDataModel()  {  }  public void Validate()  {  if (ProductId.IsEmpty())  throw new ValidationException("Field ProductId is empty");  if (!ProductId.IsGuid())  throw new ValidationException("The value in the field ProductId is not a Guid");  if (DetailId.IsEmpty())  throw new ValidationException("Field DetailId is empty");  if (!DetailId.IsGuid())  throw new ValidationException("The value in the field DetailId is not a Guid");  if (Quantity <= 0)  throw new ValidationException("Field Quantity is less than or equal to 0");  }  }  // ProductDataModel.cs  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.Infrastructure;  namespace GoToWorkContracts.DataModels;  public class ProductDataModel : IValidation  {  public string Id { get; set; }  public string? MachineId { get; set; }  public string Name { get; set; }  public DateTime CreationDate { get; set; }  public List<DetailProductDataModel>? Details { get; set; }  public string MachineName => Machine?.Model ?? string.Empty;  public MachineDataModel? Machine { get; set; }  public ProductDataModel(string id, string? machineId, string name, DateTime creationDate,  List<DetailProductDataModel> details)  {  Id = id;  MachineId = machineId;  Name = name;  CreationDate = creationDate;  Details = details;  }  public ProductDataModel(string id, string? machineId, string name, DateTime creationDate,  List<DetailProductDataModel> details, MachineDataModel? machine) : this(id, machineId, name, creationDate,  details)  {  Machine = machine;  }  public ProductDataModel()  {  }  public void Validate()  {  if (Id.IsEmpty())  throw new ValidationException("Field Id is empty");  if (!Id.IsGuid())  throw new ValidationException("The value in the field Id is not a Guid");  if (Name.IsEmpty())  throw new ValidationException("Field Name is empty");  if (MachineId is not null)  {  if (MachineId.IsEmpty())  throw new ValidationException("Field MachineId is empty");  if (!MachineId.IsGuid())  throw new ValidationException("The value in the field MachineId is not a Guid");  }  }  }  // EmployeeDataModel.cs  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.Infrastructure;  namespace GoToWorkContracts.DataModels;  public class EmployeeDataModel : IValidation  {  public string Id { get; set; }  public string FullName { get; set; }  public EmployeeDataModel(string id, string fullName)  {  Id = id;  FullName = fullName;  }  public EmployeeDataModel()  {  }  public void Validate()  {  if (Id.IsEmpty())  throw new ValidationException("Field Id is empty");  if (!Id.IsGuid())  throw new ValidationException("The value in the field Id is not a Guid");  if (FullName.IsEmpty())  throw new ValidationException("Field FullName is empty");  }  }  // DetailDataModel.cs  using GoToWorkContracts.Enums;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.Infrastructure;  namespace GoToWorkContracts.DataModels;  public class DetailDataModel : IValidation  {  public string Id { get; set; }  public string Name { get; set; }  public MaterialType Material { get; set; }  public DateTime CreationDate { get; set; }  public DetailDataModel(string id, string name, MaterialType material, DateTime creationDate)  {  Id = id;  Name = name;  Material = material;  CreationDate = creationDate.ToUniversalTime();  }  public DetailDataModel()  {  }  public void Validate()  {  if (Id.IsEmpty())  throw new ValidationException("Field Id is empty");  if (!Id.IsGuid())  throw new ValidationException("The value in the field Id is not a Guid");  if (Name.IsEmpty())  throw new ValidationException("Field Name is empty");  if (Material == MaterialType.None)  throw new ValidationException("Field Material is empty");  if (CreationDate > DateTime.UtcNow)  throw new ValidationException("The value in the field Id is not a valid Date");  }  }  // DetailProductionDataModel.cs  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.Infrastructure;  namespace GoToWorkContracts.DataModels;  public class DetailProductionDataModel : IValidation  {  public string DetailId { get; set; }  public string ProductionId { get; set; }  public string DetailName => \_detail?.Name ?? string.Empty;  public string ProductionName => \_production?.Name ?? string.Empty;  private readonly DetailDataModel? \_detail;  private readonly ProductionDataModel? \_production;  public DetailProductionDataModel(string detailId, string productionId)  {  DetailId = detailId;  ProductionId = productionId;  }  public DetailProductionDataModel(string detailId, string productionId, DetailDataModel? detail,  ProductionDataModel? production) : this(detailId, productionId)  {  \_detail = detail;  \_production = production;  }  public DetailProductionDataModel()  {  }  public void Validate()  {  if (ProductionId.IsEmpty())  throw new ValidationException("Field ProductionId is empty");  if (!ProductionId.IsGuid())  throw new ValidationException("The value in the field ProductionId is not a Guid");  if (DetailId.IsEmpty())  throw new ValidationException("Field DetailId is empty");  if (!DetailId.IsGuid())  throw new ValidationException("The value in the field DetailId is not a Guid");  }  }  // WorkshopDataModel.cs  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.Infrastructure;  namespace GoToWorkContracts.DataModels;  public class WorkshopDataModel : IValidation  {  public string Id { get; set; }  public string? ProductionId { get; set; }  public string Address { get; set; }  public List<EmployeeWorkshopDataModel>? Employees { get; set; }  public string ProductionName => Production?.Name ?? string.Empty;  public ProductionDataModel? Production { get; set; }  public WorkshopDataModel(string id, string? productionId, string address, List<EmployeeWorkshopDataModel> employees)  {  Id = id;  ProductionId = productionId;  Address = address;  Employees = employees;  }  public WorkshopDataModel(string id, string? productionId, string address, List<EmployeeWorkshopDataModel> employees,  ProductionDataModel? production) : this(id, productionId, address, employees)  {  Production = production;  }  public WorkshopDataModel()  {  }  public void Validate()  {  if (Id.IsEmpty())  throw new ValidationException("Field Id is empty");  if (!Id.IsGuid())  throw new ValidationException("The value in the field Id is not a Guid");  if (Address.IsEmpty())  throw new ValidationException("Field Address is empty");  if (ProductionId is not null)  {  if (ProductionId.IsEmpty())  throw new ValidationException("Field ProductionId is empty");  if (!ProductionId.IsGuid())  throw new ValidationException("The value in the field ProductionId is not a Guid");  }  }  }  // EmployeeMachineDataModel.cs  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.Infrastructure;  namespace GoToWorkContracts.DataModels;  public class EmployeeMachineDataModel : IValidation  {  public string EmployeeId { get; set; }  public string MachineId { get; set; }  public string EmployeeName => \_employee?.FullName ?? string.Empty;  public string MachineName => \_machine?.Model ?? string.Empty;  private readonly EmployeeDataModel? \_employee;  private readonly MachineDataModel? \_machine;  public EmployeeMachineDataModel(string employeeId, string machineId)  {  EmployeeId = employeeId;  MachineId = machineId;  }  public EmployeeMachineDataModel(string employeeId, string machineId, EmployeeDataModel employee,  MachineDataModel machine) : this(employeeId, machineId)  {  \_employee = employee;  \_machine = machine;  }  public EmployeeMachineDataModel()  {  }  public void Validate()  {  if (EmployeeId.IsEmpty())  throw new ValidationException("Field EmployeeId is empty");  if (!EmployeeId.IsGuid())  throw new ValidationException("The value in the field EmployeeId is not a Guid");  if (MachineId.IsEmpty())  throw new ValidationException("Field MachineId is empty");  if (!MachineId.IsGuid())  throw new ValidationException("The value in the field MachineId is not a Guid");  }  }  // MachineDataModel.cs  using GoToWorkContracts.Enums;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.Infrastructure;  namespace GoToWorkContracts.DataModels;  public class MachineDataModel : IValidation  {  public string Id { get; set; }  public string Model { get; set; }  public MachineType Type { get; set; }  public List<EmployeeMachineDataModel>? Employees { get; set; }  public List<ProductDataModel>? Products { get; set; }  public MachineDataModel(string id, string model, MachineType type, List<EmployeeMachineDataModel> employees,  List<ProductDataModel>? products)  {  Id = id;  Model = model;  Type = type;  Employees = employees;  Products = products;  }  public MachineDataModel()  {  }  public void Validate()  {  if (Id.IsEmpty())  throw new ValidationException("Field Id is empty");  if (!Id.IsGuid())  throw new ValidationException("The value in the field Id is not a Guid");  if (Model.IsEmpty())  throw new ValidationException("Field Model is empty");  if (Type == MachineType.None)  throw new ValidationException("Field Machine is empty");  }  }  // ProductionDataModel.cs  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.Extensions;  using GoToWorkContracts.Infrastructure;  namespace GoToWorkContracts.DataModels;  public class ProductionDataModel : IValidation  {  public string Id { get; set; }  public string Name { get; set; }  public List<DetailProductionDataModel>? Details { get; set; }  public List<WorkshopDataModel>? Workshops { get; set; }  public ProductionDataModel(string id, string name, List<DetailProductionDataModel> details,  List<WorkshopDataModel>? workshops)  {  Id = id;  Name = name;  Details = details;  Workshops = workshops;  }  public ProductionDataModel()  {  }  public void Validate()  {  if (Id.IsEmpty())  throw new ValidationException("Field Id is empty");  if (!Id.IsGuid())  throw new ValidationException("The value in the field Id is not a Guid");  if (Name.IsEmpty())  throw new ValidationException("Field Name is empty");  }  }  // GoToWorkContracts.csproj.nuget.g.targets  ﻿<?xml version="1.0" encoding="utf-8" standalone="no"?>  <Project ToolsVersion="14.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">  <ImportGroup Condition=" '$(ExcludeRestorePackageImports)' != 'true' ">  <Import Project="$(NuGetPackageRoot)microsoft.extensions.options/8.0.2/buildTransitive/net6.0/Microsoft.Extensions.Options.targets" Condition="Exists('$(NuGetPackageRoot)microsoft.extensions.options/8.0.2/buildTransitive/net6.0/Microsoft.Extensions.Options.targets')" />  <Import Project="$(NuGetPackageRoot)microsoft.extensions.logging.abstractions/8.0.2/buildTransitive/net6.0/Microsoft.Extensions.Logging.Abstractions.targets" Condition="Exists('$(NuGetPackageRoot)microsoft.extensions.logging.abstractions/8.0.2/buildTransitive/net6.0/Microsoft.Extensions.Logging.Abstractions.targets')" />  </ImportGroup>  </Project>  // GoToWorkContracts.AssemblyInfo.cs  //------------------------------------------------------------------------------  // <auto-generated>  // This code was generated by a tool.  //  // Changes to this file may cause incorrect behavior and will be lost if  // the code is regenerated.  // </auto-generated>  //------------------------------------------------------------------------------  using System;  using System.Reflection;  [assembly: System.Reflection.AssemblyCompanyAttribute("GoToWorkContracts")]  [assembly: System.Reflection.AssemblyConfigurationAttribute("Release")]  [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]  [assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+8753cd23468d9b41857b154ffbb5889134ac4e4c")]  [assembly: System.Reflection.AssemblyProductAttribute("GoToWorkContracts")]  [assembly: System.Reflection.AssemblyTitleAttribute("GoToWorkContracts")]  [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("GoToWorkApi")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("GoToWorkDatabase")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("GoToWorkBusinessLogic")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("DynamicProxyGenAssembly2")]  // Generated by the MSBuild WriteCodeFragment class.  // .NETCoreApp,Version=v9.0.AssemblyAttributes.cs  // <autogenerated />  using System;  using System.Reflection;  [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v9.0", FrameworkDisplayName = ".NET 9.0")]  // GoToWorkContracts.GlobalUsings.g.cs  // <auto-generated/>  global using global::System;  global using global::System.Collections.Generic;  global using global::System.IO;  global using global::System.Linq;  global using global::System.Net.Http;  global using global::System.Threading;  global using global::System.Threading.Tasks;  // GoToWorkContracts.AssemblyInfo.cs  //------------------------------------------------------------------------------  // <auto-generated>  // This code was generated by a tool.  //  // Changes to this file may cause incorrect behavior and will be lost if  // the code is regenerated.  // </auto-generated>  //------------------------------------------------------------------------------  using System;  using System.Reflection;  [assembly: System.Reflection.AssemblyCompanyAttribute("GoToWorkContracts")]  [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]  [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]  [assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+8753cd23468d9b41857b154ffbb5889134ac4e4c")]  [assembly: System.Reflection.AssemblyProductAttribute("GoToWorkContracts")]  [assembly: System.Reflection.AssemblyTitleAttribute("GoToWorkContracts")]  [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("GoToWorkApi")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("GoToWorkDatabase")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("GoToWorkBusinessLogic")]  [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("DynamicProxyGenAssembly2")]  // Generated by the MSBuild WriteCodeFragment class.  // .NETCoreApp,Version=v9.0.AssemblyAttributes.cs  // <autogenerated />  using System;  using System.Reflection;  [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v9.0", FrameworkDisplayName = ".NET 9.0")]  // GoToWorkContracts.GlobalUsings.g.cs  // <auto-generated/>  global using global::System;  global using global::System.Collections.Generic;  global using global::System.IO;  global using global::System.Linq;  global using global::System.Net.Http;  global using global::System.Threading;  global using global::System.Threading.Tasks;  // IUserStorageContract.cs  using GoToWorkContracts.DataModels;  namespace GoToWorkContracts.StoragesContracts;  public interface IUserStorageContract  {  List<UserDataModel> GetList();  UserDataModel? GetElementById(string id);  UserDataModel? GetElementByLogin(string login);  UserDataModel? GetElementByEmail(string email);  void AddElement(UserDataModel userDataModel);  void UpdElement(UserDataModel userDataModel);  void DelElement(string id);  }  // IMachineStorageContract.cs  using GoToWorkContracts.DataModels;  namespace GoToWorkContracts.StoragesContracts;  public interface IMachineStorageContract  {  List<MachineDataModel> GetList();  MachineDataModel? GetElementById(string id);  MachineDataModel? GetElementByModel(string model);  void AddElement(MachineDataModel machineDataModel);  void UpdElement(MachineDataModel machineDataModel);  void DelElement(string id);  }  // IWorkshopStorageContract.cs  using GoToWorkContracts.DataModels;  namespace GoToWorkContracts.StoragesContracts;  public interface IWorkshopStorageContract  {  List<WorkshopDataModel> GetList(string? productionId = null);  WorkshopDataModel? GetElementById(string id);  public WorkshopDataModel? GetElementByAddress(string address);  void AddElement(WorkshopDataModel workshopDataModel);  void UpdElement(WorkshopDataModel workshopDataModel);  void DelElement(string id);  }  // IDetailStorageContract.cs  using GoToWorkContracts.DataModels;  namespace GoToWorkContracts.StoragesContracts;  public interface IDetailStorageContract  {  public List<DetailDataModel> GetList(DateTime? startDate = null, DateTime? endDate = null);  DetailDataModel? GetElementById(string id);  DetailDataModel? GetElementByName(string name);  void AddElement(DetailDataModel detailDataModel);  void UpdElement(DetailDataModel detailDataModel);  void DelElement(string id);  }  // IEmployeeStorageContract.cs  using GoToWorkContracts.DataModels;  namespace GoToWorkContracts.StoragesContracts;  public interface IEmployeeStorageContract  {  List<EmployeeDataModel> GetList();  EmployeeDataModel? GetElementById(string id);  void AddElement(EmployeeDataModel employeeDataModel);  void UpdElement(EmployeeDataModel employeeDataModel);  void DelElement(string id);  }  // IProductStorageContract.cs  using GoToWorkContracts.DataModels;  namespace GoToWorkContracts.StoragesContracts;  public interface IProductStorageContract  {  List<ProductDataModel> GetList(DateTime? startDate = null, DateTime? endDate = null, string? machineId = null);  ProductDataModel? GetElementById(string id);  ProductDataModel? GetElementByName(string name);  void AddElement(ProductDataModel productDataModel);  void UpdElement(ProductDataModel productDataModel);  void DelElement(string id);  }  // IProductionStorageContract.cs  using GoToWorkContracts.DataModels;  namespace GoToWorkContracts.StoragesContracts;  public interface IProductionStorageContract  {  List<ProductionDataModel> GetList();  ProductionDataModel? GetElementById(string id);  ProductionDataModel? GetElementByName(string name);  void AddElement(ProductionDataModel productionDataModel);  void UpdElement(ProductionDataModel productionDataModel);  void DelElement(string id);  }  // IConfigurationDatabase.cs  namespace GoToWorkContracts.Infrastructure;  public interface IConfigurationDatabase  {  string ConnectionString { get; }  }  // OperationResponse.cs  using System.Net;  using Microsoft.AspNetCore.Http;  using Microsoft.AspNetCore.Mvc;  namespace GoToWorkContracts.Infrastructure;  public class OperationResponse  {  protected HttpStatusCode StatusCode { get; set; }  protected object? Result { get; set; }  protected string? FileName { get; set; }  public IActionResult GetResponse(HttpRequest request, HttpResponse response)  {  ArgumentNullException.ThrowIfNull(request);  ArgumentNullException.ThrowIfNull(response);  response.StatusCode = (int)StatusCode;  return Result switch  {  null => new StatusCodeResult((int)StatusCode),  Stream stream => new FileStreamResult(stream, "application/octet-stream") { FileDownloadName = FileName },  \_ => new ObjectResult(Result)  };  }  protected static TResult OK<TResult, TData>(TData data) where TResult : OperationResponse, new() =>  new() { StatusCode = HttpStatusCode.OK, Result = data };  protected static TResult OK<TResult, TData>(TData data, string filename) where TResult : OperationResponse, new() =>  new() { StatusCode = HttpStatusCode.OK, Result = data, FileName = filename };  protected static TResult NoContent<TResult>() where TResult : OperationResponse, new() =>  new() { StatusCode = HttpStatusCode.NoContent };  protected static TResult BadRequest<TResult>(string? errorMessage = null)  where TResult : OperationResponse, new() =>  new() { StatusCode = HttpStatusCode.BadRequest, Result = errorMessage };  protected static TResult NotFound<TResult>(string? errorMessage = null)  where TResult : OperationResponse, new() =>  new() { StatusCode = HttpStatusCode.NotFound, Result = errorMessage };  protected static TResult InternalServerError<TResult>(string? errorMessage = null)  where TResult : OperationResponse, new() =>  new() { StatusCode = HttpStatusCode.InternalServerError, Result = errorMessage };  protected static TResult Unauthorized<TResult>(string? errorMessage = null)  where TResult : OperationResponse, new() =>  new() { StatusCode = HttpStatusCode.Unauthorized, Result = errorMessage };  }  // IValidation.cs  namespace GoToWorkContracts.Infrastructure;  public interface IValidation  {  void Validate();  }  // Program.cs  using GoToWorkApi;  using GoToWorkApi.Adapters;  using GoToWorkApi.Infrastructure;  using GoToWorkBusinessLogic.Implementations;  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.Infrastructure;  using GoToWorkContracts.StoragesContracts;  using GoToWorkDatabase;  using GoToWorkDatabase.Implementations;  using Microsoft.AspNetCore.Authentication.JwtBearer;  using Microsoft.AspNetCore.CookiePolicy;  using Microsoft.EntityFrameworkCore;  using Microsoft.IdentityModel.Tokens;  using Serilog;  var builder = WebApplication.CreateBuilder(args);  // Add services to the container.  // Logging  using var loggerFactory = new LoggerFactory();  loggerFactory.AddSerilog(new LoggerConfiguration().ReadFrom.Configuration(builder.Configuration).CreateLogger());  builder.Services.AddSingleton(loggerFactory.CreateLogger("Any"));  // DbContext  builder.Services.AddSingleton<IConfigurationDatabase, ConfigurationDatabase>();  builder.Services.AddDbContext<GoToWorkDbContext>(options =>  options.UseNpgsql(builder.Configuration.GetConnectionString("DefaultConnection")));  // Dependency Injection  // Storages  builder.Services.AddScoped<IDetailStorageContract, DetailStorageContract>();  builder.Services.AddScoped<IEmployeeStorageContract, EmployeeStorageContract>();  builder.Services.AddScoped<IMachineStorageContract, MachineStorageContract>();  builder.Services.AddScoped<IProductStorageContract, ProductStorageContract>();  builder.Services.AddScoped<IProductionStorageContract, ProductionStorageContract>();  builder.Services.AddScoped<IUserStorageContract, UserStorageContract>();  builder.Services.AddScoped<IWorkshopStorageContract, WorkshopStorageContract>();  // Business Logic  builder.Services.AddScoped<IDetailBusinessLogicContract, DetailBusinessLogicContract>();  builder.Services.AddScoped<IEmployeeBusinessLogicContract, EmployeeBusinessLogicContract>();  builder.Services.AddScoped<IMachineBusinessLogicContract, MachineBusinessLogicContract>();  builder.Services.AddScoped<IProductBusinessLogicContract, ProductBusinessLogicContract>();  builder.Services.AddScoped<IProductionBusinessLogicContract, ProductionBusinessLogicContract>();  builder.Services.AddScoped<IUserBusinessLogicContract, UserBusinessLogicContract>();  builder.Services.AddScoped<IWorkshopBusinessLogicContract, WorkshopBusinessLogicContract>();  builder.Services.AddScoped<IReportContract, ReportContract>();  // Adapters  builder.Services.AddScoped<IDetailAdapter, DetailAdapter>();  builder.Services.AddScoped<IEmployeeAdapter, EmployeeAdapter>();  builder.Services.AddScoped<IMachineAdapter, MachineAdapter>();  builder.Services.AddScoped<IProductAdapter, ProductAdapter>();  builder.Services.AddScoped<IProductionAdapter, ProductionAdapter>();  builder.Services.AddScoped<IUserAdapter, UserAdapter>();  builder.Services.AddScoped<IWorkshopAdapter, WorkshopAdapter>();  builder.Services.AddScoped<IReportAdapter, ReportAdapter>();  builder.Services.AddControllers();  builder.Services.AddAuthorization();  builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)  .AddJwtBearer(options =>  {  options.TokenValidationParameters = new TokenValidationParameters  {  ValidateIssuer = true,  ValidIssuer = AuthOptions.Issuer,  ValidateAudience = true,  ValidAudience = AuthOptions.Audience,  ValidateLifetime = true,  IssuerSigningKey = AuthOptions.GetSymmetricSecurityKey(),  ValidateIssuerSigningKey = true  };  });  // Learn more about configuring OpenAPI at https://aka.ms/aspnet/openapi  builder.Services.AddOpenApi();  var app = builder.Build();  // Configure the HTTP request pipeline.  if (app.Environment.IsDevelopment()) app.MapOpenApi();  if (app.Environment.IsProduction())  {  using var scope = app.Services.CreateScope();  var dbContext = scope.ServiceProvider.GetRequiredService<GoToWorkDbContext>();  if (dbContext.Database.CanConnect())  {  dbContext.Database.EnsureCreated();  dbContext.Database.Migrate();  }  }  app.UseCors("AllowFrontend");  app.UseHttpsRedirection();  app.UseCookiePolicy(new CookiePolicyOptions  {  HttpOnly = HttpOnlyPolicy.Always,  Secure = app.Environment.IsProduction() ? CookieSecurePolicy.Always : CookieSecurePolicy.None  });  app.UseAuthentication();  app.UseAuthorization();  app.MapControllers();  app.Run();  // AuthOptions.cs  using System.Text;  using Microsoft.IdentityModel.Tokens;  namespace GoToWorkApi;  public static class AuthOptions  {  public const string Issuer = "GoToWorkAuthServer";  public const string Audience = "GoToWorkAuthClient";  private const string Key = "GoToWorkCourseWorkAuthenticationSecretKey";  public static SymmetricSecurityKey GetSymmetricSecurityKey()  {  return new SymmetricSecurityKey(Encoding.UTF8.GetBytes(Key));  }  }  // WorkshopsController.cs  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.BindingModels;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  namespace GoToWorkApi.Controllers;  [Authorize]  [Route("api/[controller]/[action]")]  [ApiController]  [Produces("application/json")]  public class WorkshopsController : ControllerBase  {  private readonly IWorkshopAdapter \_adapter;  public WorkshopsController(IWorkshopAdapter adapter)  {  \_adapter = adapter;  }  [HttpGet]  public IActionResult GetAllRecords()  {  return \_adapter.GetList().GetResponse(Request, Response);  }  [HttpGet("{productionId}")]  public IActionResult GetListByProduction(string productionId)  {  return \_adapter.GetListByProduction(productionId).GetResponse(Request, Response);  }  [HttpGet("{data}")]  public IActionResult GetRecord(string data)  {  return \_adapter.GetElement(data).GetResponse(Request, Response);  }  [HttpPost]  public IActionResult Create([FromBody] WorkshopBindingModel model)  {  return \_adapter.CreateWorkshop(model).GetResponse(Request, Response);  }  [HttpPut]  public IActionResult Update([FromBody] WorkshopBindingModel model)  {  return \_adapter.UpdateWorkshop(model).GetResponse(Request, Response);  }  [HttpDelete("{id}")]  public IActionResult Delete(string id)  {  return \_adapter.DeleteWorkshop(id).GetResponse(Request, Response);  }  }  // ProductionsController.cs  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.BindingModels;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  namespace GoToWorkApi.Controllers;  [Authorize]  [Route("api/[controller]/[action]")]  [ApiController]  [Produces("application/json")]  public class ProductionsController : ControllerBase  {  private readonly IProductionAdapter \_adapter;  public ProductionsController(IProductionAdapter adapter)  {  \_adapter = adapter;  }  [HttpGet]  public IActionResult GetAllRecords()  {  return \_adapter.GetList().GetResponse(Request, Response);  }  [HttpGet("{data}")]  public IActionResult GetRecord(string data)  {  return \_adapter.GetElement(data).GetResponse(Request, Response);  }  [HttpPost]  public IActionResult Create([FromBody] ProductionBindingModel model)  {  return \_adapter.CreateProduction(model).GetResponse(Request, Response);  }  [HttpPut]  public IActionResult Update([FromBody] ProductionBindingModel model)  {  return \_adapter.UpdateProduction(model).GetResponse(Request, Response);  }  [HttpDelete("{id}")]  public IActionResult Delete(string id)  {  return \_adapter.DeleteProduction(id).GetResponse(Request, Response);  }  }  // AuthController.cs  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.BindingModels;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  namespace GoToWorkApi.Controllers;  [AllowAnonymous]  [Route("api/[controller]/[action]")]  [ApiController]  [Produces("application/json")]  public class AuthController(IUserAdapter userAdapter) : ControllerBase  {  [HttpPost]  public IActionResult Register([FromBody] UserRegisterBindingModel model)  {  return userAdapter.Register(model).GetResponse(Request, Response);  }  [HttpPost]  public IActionResult Login([FromBody] UserLoginBindingModel model)  {  return userAdapter.Login(model).GetResponse(Request, Response);  }  }  // ReportsController.cs  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.BindingModels;  using GoToWorkContracts.ViewModels;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  namespace GoToWorkApi.Controllers;  [Authorize]  [Route("api/[controller]/[action]")]  [ApiController]  public class ReportsController(IReportAdapter adapter) : ControllerBase  {  [HttpPost]  public async Task<IActionResult> GetWorkshopsReportXlsx([FromBody] WorkshopsReportBindingModel workshopsReport,  CancellationToken ct)  {  return (await adapter.CreateXlsxDocumentWorkshopsByDetailsAsync(workshopsReport, ct))  .GetResponse(Request, Response);  }  [HttpPost]  public async Task<IActionResult> GetWorkshopsReportDocx([FromBody] WorkshopsReportBindingModel workshopsReport,  CancellationToken ct)  {  return (await adapter.CreateDocxDocumentWorkshopsByDetailsAsync(workshopsReport, ct))  .GetResponse(Request, Response);  }  [HttpPost]  public async Task<IActionResult> GetDetailsReportPdf([FromBody] DetailsReportBindingModel detailsReport,  CancellationToken ct)  {  return (await adapter.CreatePdfDocumentDetailsByMachinesAndProductionsAsync(detailsReport, ct))  .GetResponse(Request, Response);  }  [HttpPost]  public async Task<IActionResult> GetDetailsReportPdfEmail([FromBody] DetailsReportBindingModel model,  CancellationToken ct)  {  return (await adapter.SendPdfDocumentDetailsByMachinesAndProductionsEmailAsync(model, ct))  .GetResponse(Request, Response);  }  }  // MachinesController.cs  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.BindingModels;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  namespace GoToWorkApi.Controllers;  [Authorize]  [Route("api/[controller]/[action]")]  [ApiController]  [Produces("application/json")]  public class MachinesController : ControllerBase  {  private readonly IMachineAdapter \_adapter;  public MachinesController(IMachineAdapter adapter)  {  \_adapter = adapter;  }  [HttpGet]  public IActionResult GetAllRecords()  {  return \_adapter.GetList().GetResponse(Request, Response);  }  [HttpGet("{data}")]  public IActionResult GetRecord(string data)  {  return \_adapter.GetElement(data).GetResponse(Request, Response);  }  [HttpPost]  public IActionResult Create([FromBody] MachineBindingModel model)  {  return \_adapter.CreateMachine(model).GetResponse(Request, Response);  }  [HttpPut]  public IActionResult Update([FromBody] MachineBindingModel model)  {  return \_adapter.UpdateMachine(model).GetResponse(Request, Response);  }  [HttpDelete("{id}")]  public IActionResult Delete(string id)  {  return \_adapter.DeleteMachine(id).GetResponse(Request, Response);  }  }  // DetailsController.cs  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.BindingModels;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  namespace GoToWorkApi.Controllers;  [Authorize]  [Route("api/[controller]/[action]")]  [ApiController]  [Produces("application/json")]  public class DetailsController(IDetailAdapter adapter) : ControllerBase  {  [HttpGet]  public IActionResult GetAllRecords()  {  return adapter.GetList().GetResponse(Request, Response);  }  [HttpGet("{data}")]  public IActionResult GetRecord(string data)  {  return adapter.GetElement(data).GetResponse(Request, Response);  }  [HttpPost]  public IActionResult Register([FromBody] DetailBindingModel model)  {  return adapter.CreateDetail(model).GetResponse(Request, Response);  }  [HttpPut]  public IActionResult ChangeInfo([FromBody] DetailBindingModel model)  {  return adapter.UpdateDetail(model).GetResponse(Request, Response);  }  [HttpDelete("{id}")]  public IActionResult Delete(string id)  {  return adapter.DeleteDetail(id).GetResponse(Request, Response);  }  }  // EmployeesController.cs  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.BindingModels;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  namespace GoToWorkApi.Controllers;  [Authorize]  [Route("api/[controller]/[action]")]  [ApiController]  [Produces("application/json")]  public class EmployeesController(IEmployeeAdapter adapter) : ControllerBase  {  [HttpGet]  public IActionResult GetAllRecords()  {  return adapter.GetList().GetResponse(Request, Response);  }  [HttpGet("{data}")]  public IActionResult GetRecord(string data)  {  return adapter.GetElement(data).GetResponse(Request, Response);  }  [HttpPost]  public IActionResult Create([FromBody] EmployeeBindingModel model)  {  return adapter.CreateEmployee(model).GetResponse(Request, Response);  }  [HttpPut]  public IActionResult Update([FromBody] EmployeeBindingModel model)  {  return adapter.UpdateEmployee(model).GetResponse(Request, Response);  }  [HttpDelete("{id}")]  public IActionResult Delete(string id)  {  return adapter.DeleteEmployee(id).GetResponse(Request, Response);  }  }  // ProductsController.cs  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.BindingModels;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  namespace GoToWorkApi.Controllers;  [Authorize]  [Route("api/[controller]/[action]")]  [ApiController]  [Produces("application/json")]  public class ProductsController : ControllerBase  {  private readonly IProductAdapter \_adapter;  public ProductsController(IProductAdapter adapter)  {  \_adapter = adapter;  }  [HttpGet]  public IActionResult GetAllRecords()  {  return \_adapter.GetList().GetResponse(Request, Response);  }  [HttpGet("{machineId}")]  public IActionResult GetListByMachine(string machineId)  {  return \_adapter.GetListByMachine(machineId).GetResponse(Request, Response);  }  [HttpGet]  public IActionResult GetListByCreationDate(DateTime from, DateTime to)  {  return \_adapter.GetListByCreationDate(from, to).GetResponse(Request, Response);  }  [HttpGet("{data}")]  public IActionResult GetRecord(string data)  {  return \_adapter.GetElement(data).GetResponse(Request, Response);  }  [HttpPost]  public IActionResult Create([FromBody] ProductBindingModel model)  {  return \_adapter.CreateProduct(model).GetResponse(Request, Response);  }  [HttpPut]  public IActionResult Update([FromBody] ProductBindingModel model)  {  return \_adapter.UpdateProduct(model).GetResponse(Request, Response);  }  [HttpDelete("{id}")]  public IActionResult Delete(string id)  {  return \_adapter.DeleteProduct(id).GetResponse(Request, Response);  }  }  // UsersController.cs  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.BindingModels;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  namespace GoToWorkApi.Controllers;  [Authorize]  [Route("api/[controller]")]  [ApiController]  [Produces("application/json")]  public class UsersController(IUserAdapter adapter) : ControllerBase  {  [HttpGet]  public IActionResult GetAllRecords()  {  return adapter.GetList().GetResponse(Request, Response);  }  [HttpGet("{data}")]  public IActionResult GetRecord(string data)  {  return adapter.GetElement(data).GetResponse(Request, Response);  }  [HttpPut]  public IActionResult Update([FromBody] UserBindingModel model)  {  return adapter.UpdateUser(model).GetResponse(Request, Response);  }  [HttpDelete("{id}")]  public IActionResult Delete(string id)  {  return adapter.DeleteUser(id).GetResponse(Request, Response);  }  }  // MachineAdapter.cs  using AutoMapper;  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.ViewModels;  namespace GoToWorkApi.Adapters;  public class MachineAdapter : IMachineAdapter  {  private readonly ILogger<MachineAdapter> \_logger;  private readonly IMachineBusinessLogicContract \_machineBusinessLogic;  private readonly Mapper \_mapper;  public MachineAdapter(IMachineBusinessLogicContract machineBusinessLogic, ILogger<MachineAdapter> logger)  {  \_machineBusinessLogic = machineBusinessLogic;  \_logger = logger;  var config = new MapperConfiguration(cfg =>  {  cfg.CreateMap<MachineBindingModel, MachineDataModel>();  cfg.CreateMap<MachineDataModel, MachineViewModel>();  cfg.CreateMap<ProductBindingModel, ProductDataModel>();  cfg.CreateMap<ProductDataModel, ProductViewModel>();  cfg.CreateMap<EmployeeBindingModel, EmployeeDataModel>();  cfg.CreateMap<EmployeeDataModel, EmployeeViewModel>();  cfg.CreateMap<EmployeeMachineBindingModel, EmployeeMachineDataModel>();  cfg.CreateMap<EmployeeMachineDataModel, EmployeeMachineViewModel>();  });  \_mapper = new Mapper(config);  }  public MachineOperationResponse GetList()  {  try  {  return MachineOperationResponse.OK([  ..\_machineBusinessLogic.GetAllMachines()  .Select(x => \_mapper.Map<MachineDataModel, MachineViewModel>(x))  ]);  }  catch (NullListException)  {  \_logger.LogError("NullListException");  return MachineOperationResponse.NotFound("The list is not initialized");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return MachineOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return MachineOperationResponse.InternalServerError(ex.Message);  }  }  public MachineOperationResponse GetElement(string data)  {  try  {  return MachineOperationResponse.OK(  \_mapper.Map<MachineViewModel>(\_machineBusinessLogic.GetMachineByData(data)));  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return MachineOperationResponse.BadRequest("Data is empty");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return MachineOperationResponse.NotFound($"Not found element by data {data}");  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return MachineOperationResponse.BadRequest($"Element by data: {data} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return MachineOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return MachineOperationResponse.InternalServerError(ex.Message);  }  }  public MachineOperationResponse CreateMachine(MachineBindingModel machineModel)  {  try  {  machineModel.Id = Guid.NewGuid().ToString();  \_machineBusinessLogic.InsertMachine(\_mapper.Map<MachineDataModel>(machineModel));  return MachineOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return MachineOperationResponse.BadRequest("Data is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return MachineOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementExistsException ex)  {  \_logger.LogError(ex, "ElementExistsException");  return MachineOperationResponse.BadRequest(ex.Message);  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return MachineOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return MachineOperationResponse.InternalServerError(ex.Message);  }  }  public MachineOperationResponse UpdateMachine(MachineBindingModel machineModel)  {  try  {  \_machineBusinessLogic.UpdateMachine(\_mapper.Map<MachineDataModel>(machineModel));  return MachineOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return MachineOperationResponse.BadRequest("Data is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return MachineOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return MachineOperationResponse.BadRequest($"Not found element by Id {machineModel.Id}");  }  catch (ElementExistsException ex)  {  \_logger.LogError(ex, "ElementExistsException");  return MachineOperationResponse.BadRequest(ex.Message);  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return MachineOperationResponse.BadRequest($"Element by id: {machineModel.Id} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return MachineOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return MachineOperationResponse.InternalServerError(ex.Message);  }  }  public MachineOperationResponse DeleteMachine(string id)  {  try  {  \_machineBusinessLogic.DeleteMachine(id);  return MachineOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return MachineOperationResponse.BadRequest("Id is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return MachineOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return MachineOperationResponse.BadRequest($"Not found element by id: {id}");  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return MachineOperationResponse.BadRequest($"Element by id: {id} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return MachineOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return MachineOperationResponse.InternalServerError(ex.Message);  }  }  }  // ProductionAdapter.cs  using AutoMapper;  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.ViewModels;  namespace GoToWorkApi.Adapters;  public class ProductionAdapter : IProductionAdapter  {  private readonly ILogger<ProductionAdapter> \_logger;  private readonly Mapper \_mapper;  private readonly IProductionBusinessLogicContract \_productionBusinessLogic;  public ProductionAdapter(IProductionBusinessLogicContract productionBusinessLogic,  ILogger<ProductionAdapter> logger)  {  \_productionBusinessLogic = productionBusinessLogic;  \_logger = logger;  var config = new MapperConfiguration(cfg =>  {  cfg.CreateMap<ProductionBindingModel, ProductionDataModel>();  cfg.CreateMap<DetailProductionBindingModel, DetailProductionDataModel>();  cfg.CreateMap<WorkshopBindingModel, WorkshopDataModel>();  cfg.CreateMap<ProductionDataModel, ProductionViewModel>();  cfg.CreateMap<DetailProductionDataModel, DetailProductionViewModel>();  cfg.CreateMap<WorkshopDataModel, WorkshopViewModel>();  });  \_mapper = new Mapper(config);  }  public ProductionOperationResponse GetList()  {  try  {  return ProductionOperationResponse.OK([  ..\_productionBusinessLogic.GetAllProductions()  .Select(x => \_mapper.Map<ProductionDataModel, ProductionViewModel>(x))  ]);  }  catch (NullListException)  {  \_logger.LogError("NullListException");  return ProductionOperationResponse.NotFound("The list is not initialized");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return ProductionOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return ProductionOperationResponse.InternalServerError(ex.Message);  }  }  public ProductionOperationResponse GetElement(string data)  {  try  {  return ProductionOperationResponse.OK(  \_mapper.Map<ProductionViewModel>(\_productionBusinessLogic.GetProductionByData(data)));  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return ProductionOperationResponse.BadRequest("Data is empty");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return ProductionOperationResponse.NotFound($"Not found element by data {data}");  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return ProductionOperationResponse.BadRequest($"Element by data: {data} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return ProductionOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return ProductionOperationResponse.InternalServerError(ex.Message);  }  }  public ProductionOperationResponse CreateProduction(ProductionBindingModel productionModel)  {  try  {  productionModel.Id = Guid.NewGuid().ToString();  \_productionBusinessLogic.InsertProduction(\_mapper.Map<ProductionDataModel>(productionModel));  return ProductionOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return ProductionOperationResponse.BadRequest("Data is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return ProductionOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementExistsException ex)  {  \_logger.LogError(ex, "ElementExistsException");  return ProductionOperationResponse.BadRequest(ex.Message);  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return ProductionOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return ProductionOperationResponse.InternalServerError(ex.Message);  }  }  public ProductionOperationResponse UpdateProduction(ProductionBindingModel productionModel)  {  try  {  \_productionBusinessLogic.UpdateProduction(\_mapper.Map<ProductionDataModel>(productionModel));  return ProductionOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return ProductionOperationResponse.BadRequest("Data is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return ProductionOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return ProductionOperationResponse.BadRequest($"Not found element by Id {productionModel.Id}");  }  catch (ElementExistsException ex)  {  \_logger.LogError(ex, "ElementExistsException");  return ProductionOperationResponse.BadRequest(ex.Message);  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return ProductionOperationResponse.BadRequest($"Element by id: {productionModel.Id} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return ProductionOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return ProductionOperationResponse.InternalServerError(ex.Message);  }  }  public ProductionOperationResponse DeleteProduction(string id)  {  try  {  \_productionBusinessLogic.DeleteProduction(id);  return ProductionOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return ProductionOperationResponse.BadRequest("Id is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return ProductionOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return ProductionOperationResponse.BadRequest($"Not found element by id: {id}");  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return ProductionOperationResponse.BadRequest($"Element by id: {id} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return ProductionOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return ProductionOperationResponse.InternalServerError(ex.Message);  }  }  }  // EmployeeAdapter.cs  using AutoMapper;  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.ViewModels;  namespace GoToWorkApi.Adapters;  public class EmployeeAdapter : IEmployeeAdapter  {  private readonly IEmployeeBusinessLogicContract \_employeeBusinessLogic;  private readonly ILogger<EmployeeAdapter> \_logger;  private readonly Mapper \_mapper;  public EmployeeAdapter(IEmployeeBusinessLogicContract employeeBusinessLogic, ILogger<EmployeeAdapter> logger)  {  \_employeeBusinessLogic = employeeBusinessLogic;  \_logger = logger;  var config = new MapperConfiguration(cfg =>  {  cfg.CreateMap<EmployeeBindingModel, EmployeeDataModel>();  cfg.CreateMap<EmployeeDataModel, EmployeeViewModel>();  });  \_mapper = new Mapper(config);  }  public EmployeeOperationResponse GetList()  {  try  {  return EmployeeOperationResponse.OK([  ..\_employeeBusinessLogic.GetAllEmployees()  .Select(x => \_mapper.Map<EmployeeDataModel, EmployeeViewModel>(x))  ]);  }  catch (NullListException)  {  \_logger.LogError("NullListException");  return EmployeeOperationResponse.NotFound("The list is not initialized");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return EmployeeOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return EmployeeOperationResponse.InternalServerError(ex.Message);  }  }  public EmployeeOperationResponse GetElement(string data)  {  try  {  return EmployeeOperationResponse.OK(  \_mapper.Map<EmployeeViewModel>(\_employeeBusinessLogic.GetEmployeeByData(data)));  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return EmployeeOperationResponse.BadRequest("Data is empty");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return EmployeeOperationResponse.NotFound($"Not found element by data {data}");  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return EmployeeOperationResponse.BadRequest($"Element by data: {data} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return EmployeeOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return EmployeeOperationResponse.InternalServerError(ex.Message);  }  }  public EmployeeOperationResponse CreateEmployee(EmployeeBindingModel employeeModel)  {  try  {  employeeModel.Id = Guid.NewGuid().ToString();  \_employeeBusinessLogic.InsertEmployee(\_mapper.Map<EmployeeDataModel>(employeeModel));  return EmployeeOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return EmployeeOperationResponse.BadRequest("Data is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return EmployeeOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementExistsException ex)  {  \_logger.LogError(ex, "ElementExistsException");  return EmployeeOperationResponse.BadRequest(ex.Message);  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return EmployeeOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return EmployeeOperationResponse.InternalServerError(ex.Message);  }  }  public EmployeeOperationResponse UpdateEmployee(EmployeeBindingModel employeeModel)  {  try  {  \_employeeBusinessLogic.UpdateEmployee(\_mapper.Map<EmployeeDataModel>(employeeModel));  return EmployeeOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return EmployeeOperationResponse.BadRequest("Data is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return EmployeeOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return EmployeeOperationResponse.BadRequest($"Not found element by Id {employeeModel.Id}");  }  catch (ElementExistsException ex)  {  \_logger.LogError(ex, "ElementExistsException");  return EmployeeOperationResponse.BadRequest(ex.Message);  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return EmployeeOperationResponse.BadRequest($"Element by id: {employeeModel.Id} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return EmployeeOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return EmployeeOperationResponse.InternalServerError(ex.Message);  }  }  public EmployeeOperationResponse DeleteEmployee(string id)  {  try  {  \_employeeBusinessLogic.DeleteEmployee(id);  return EmployeeOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return EmployeeOperationResponse.BadRequest("Id is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return EmployeeOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return EmployeeOperationResponse.BadRequest($"Not found element by id: {id}");  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return EmployeeOperationResponse.BadRequest($"Element by id: {id} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return EmployeeOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return EmployeeOperationResponse.InternalServerError(ex.Message);  }  }  }  // UserAdapter.cs  using System.IdentityModel.Tokens.Jwt;  using System.Security.Claims;  using AutoMapper;  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.ViewModels;  using Microsoft.IdentityModel.Tokens;  namespace GoToWorkApi.Adapters;  public class UserAdapter : IUserAdapter  {  private readonly ILogger<UserAdapter> \_logger;  private readonly Mapper \_mapper;  private readonly IUserBusinessLogicContract \_userBusinessLogic;  public UserAdapter(IUserBusinessLogicContract userBusinessLogic, ILogger<UserAdapter> logger)  {  \_userBusinessLogic = userBusinessLogic;  \_logger = logger;  var config = new MapperConfiguration(cfg =>  {  cfg.CreateMap<UserBindingModel, UserDataModel>();  cfg.CreateMap<UserRegisterBindingModel, UserDataModel>();  cfg.CreateMap<UserDataModel, UserViewModel>();  });  \_mapper = new Mapper(config);  }  public UserOperationResponse GetList()  {  try  {  return UserOperationResponse.OK([  ..\_userBusinessLogic.GetAllUsers()  .Select(x => \_mapper.Map<UserDataModel, UserViewModel>(x))  ]);  }  catch (NullListException)  {  \_logger.LogError("NullListException");  return UserOperationResponse.NotFound("The list is not initialized");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return UserOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return UserOperationResponse.InternalServerError(ex.Message);  }  }  public UserOperationResponse GetElement(string data)  {  try  {  return UserOperationResponse.OK(  \_mapper.Map<UserViewModel>(\_userBusinessLogic.GetUserByData(data)));  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return UserOperationResponse.BadRequest("Data is empty");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return UserOperationResponse.NotFound($"Not found element by data {data}");  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return UserOperationResponse.BadRequest($"Element by data: {data} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return UserOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return UserOperationResponse.InternalServerError(ex.Message);  }  }  public UserOperationResponse CreateUser(UserBindingModel userModel)  {  try  {  userModel.Id = Guid.NewGuid().ToString();  \_userBusinessLogic.InsertUser(\_mapper.Map<UserDataModel>(userModel));  return UserOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return UserOperationResponse.BadRequest("Data is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return UserOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementExistsException ex)  {  \_logger.LogError(ex, "ElementExistsException");  return UserOperationResponse.BadRequest(ex.Message);  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return UserOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return UserOperationResponse.InternalServerError(ex.Message);  }  }  public UserOperationResponse UpdateUser(UserBindingModel userModel)  {  try  {  \_userBusinessLogic.UpdateUser(\_mapper.Map<UserDataModel>(userModel));  return UserOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return UserOperationResponse.BadRequest("Data is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return UserOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return UserOperationResponse.BadRequest($"Not found element by Id {userModel.Id}");  }  catch (ElementExistsException ex)  {  \_logger.LogError(ex, "ElementExistsException");  return UserOperationResponse.BadRequest(ex.Message);  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return UserOperationResponse.BadRequest($"Element by id: {userModel.Id} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return UserOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return UserOperationResponse.InternalServerError(ex.Message);  }  }  public UserOperationResponse DeleteUser(string id)  {  try  {  \_userBusinessLogic.DeleteUser(id);  return UserOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return UserOperationResponse.BadRequest("Id is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return UserOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return UserOperationResponse.BadRequest($"Not found element by id: {id}");  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return UserOperationResponse.BadRequest($"Element by id: {id} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return UserOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return UserOperationResponse.InternalServerError(ex.Message);  }  }  public UserOperationResponse Register(UserRegisterBindingModel model)  {  try  {  \_userBusinessLogic.Register(\_mapper.Map<UserDataModel>(model));  return UserOperationResponse.NoContent();  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return UserOperationResponse.InternalServerError(ex.Message);  }  }  public AuthOperationResponse Login(UserLoginBindingModel model)  {  try  {  var user = \_userBusinessLogic.Login(model.Login, model.Password);  if (user == null) return AuthOperationResponse.Unauthorized("Invalid login or password");  var claims = new List<Claim>  {  new(ClaimsIdentity.DefaultNameClaimType, user.Value.login),  new(ClaimsIdentity.DefaultRoleClaimType, user.Value.role.ToString()),  new("id", user.Value.id)  };  var jwt = new JwtSecurityToken(  AuthOptions.Issuer,  AuthOptions.Audience,  claims,  expires: DateTime.UtcNow.Add(TimeSpan.FromMinutes(60)),  signingCredentials: new SigningCredentials(AuthOptions.GetSymmetricSecurityKey(),  SecurityAlgorithms.HmacSha256));  var encodedJwt = new JwtSecurityTokenHandler().WriteToken(jwt);  var tokenModel = new TokenViewModel { Token = encodedJwt };  return AuthOperationResponse.OK(tokenModel);  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return AuthOperationResponse.BadRequest(ex.Message);  }  }  }  // WorkshopAdapter.cs  using AutoMapper;  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.ViewModels;  namespace GoToWorkApi.Adapters;  public class WorkshopAdapter : IWorkshopAdapter  {  private readonly ILogger<WorkshopAdapter> \_logger;  private readonly Mapper \_mapper;  private readonly IWorkshopBusinessLogicContract \_workshopBusinessLogic;  public WorkshopAdapter(IWorkshopBusinessLogicContract workshopBusinessLogic, ILogger<WorkshopAdapter> logger)  {  \_workshopBusinessLogic = workshopBusinessLogic;  \_logger = logger;  var config = new MapperConfiguration(cfg =>  {  cfg.CreateMap<WorkshopBindingModel, WorkshopDataModel>();  cfg.CreateMap<WorkshopDataModel, WorkshopViewModel>();  cfg.CreateMap<EmployeeBindingModel, EmployeeDataModel>();  cfg.CreateMap<EmployeeDataModel, EmployeeViewModel>();  cfg.CreateMap<EmployeeWorkshopBindingModel, EmployeeWorkshopDataModel>();  cfg.CreateMap<EmployeeWorkshopDataModel, EmployeeWorkshopViewModel>();  });  \_mapper = new Mapper(config);  }  public WorkshopOperationResponse GetList()  {  try  {  return WorkshopOperationResponse.OK([  ..\_workshopBusinessLogic.GetAllWorkshops()  .Select(x => \_mapper.Map<WorkshopDataModel, WorkshopViewModel>(x))  ]);  }  catch (NullListException)  {  \_logger.LogError("NullListException");  return WorkshopOperationResponse.NotFound("The list is not initialized");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return WorkshopOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return WorkshopOperationResponse.InternalServerError(ex.Message);  }  }  public WorkshopOperationResponse GetListByProduction(string productionId)  {  try  {  return WorkshopOperationResponse.OK([  ..\_workshopBusinessLogic.GetWorkshopsByProduction(productionId)  .Select(x => \_mapper.Map<WorkshopDataModel, WorkshopViewModel>(x))  ]);  }  catch (NullListException)  {  \_logger.LogError("NullListException");  return WorkshopOperationResponse.NotFound("The list is not initialized");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return WorkshopOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return WorkshopOperationResponse.InternalServerError(ex.Message);  }  }  public WorkshopOperationResponse GetElement(string data)  {  try  {  return WorkshopOperationResponse.OK(  \_mapper.Map<WorkshopViewModel>(\_workshopBusinessLogic.GetWorkshopByData(data)));  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return WorkshopOperationResponse.BadRequest("Data is empty");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return WorkshopOperationResponse.NotFound($"Not found element by data {data}");  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return WorkshopOperationResponse.BadRequest($"Element by data: {data} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return WorkshopOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return WorkshopOperationResponse.InternalServerError(ex.Message);  }  }  public WorkshopOperationResponse CreateWorkshop(WorkshopBindingModel workshopModel)  {  try  {  workshopModel.Id = Guid.NewGuid().ToString();  \_workshopBusinessLogic.InsertWorkshop(\_mapper.Map<WorkshopDataModel>(workshopModel));  return WorkshopOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return WorkshopOperationResponse.BadRequest("Data is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return WorkshopOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementExistsException ex)  {  \_logger.LogError(ex, "ElementExistsException");  return WorkshopOperationResponse.BadRequest(ex.Message);  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return WorkshopOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return WorkshopOperationResponse.InternalServerError(ex.Message);  }  }  public WorkshopOperationResponse UpdateWorkshop(WorkshopBindingModel workshopModel)  {  try  {  \_workshopBusinessLogic.UpdateWorkshop(\_mapper.Map<WorkshopDataModel>(workshopModel));  return WorkshopOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return WorkshopOperationResponse.BadRequest("Data is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return WorkshopOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return WorkshopOperationResponse.BadRequest($"Not found element by Id {workshopModel.Id}");  }  catch (ElementExistsException ex)  {  \_logger.LogError(ex, "ElementExistsException");  return WorkshopOperationResponse.BadRequest(ex.Message);  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return WorkshopOperationResponse.BadRequest($"Element by id: {workshopModel.Id} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return WorkshopOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return WorkshopOperationResponse.InternalServerError(ex.Message);  }  }  public WorkshopOperationResponse DeleteWorkshop(string id)  {  try  {  \_workshopBusinessLogic.DeleteWorkshop(id);  return WorkshopOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return WorkshopOperationResponse.BadRequest("Id is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return WorkshopOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return WorkshopOperationResponse.BadRequest($"Not found element by id: {id}");  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return WorkshopOperationResponse.BadRequest($"Element by id: {id} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return WorkshopOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return WorkshopOperationResponse.InternalServerError(ex.Message);  }  }  }  // DetailAdapter.cs  using AutoMapper;  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.ViewModels;  namespace GoToWorkApi.Adapters;  public class DetailAdapter : IDetailAdapter  {  private readonly IDetailBusinessLogicContract \_detailBusinessLogicContract;  private readonly ILogger \_logger;  private readonly Mapper \_mapper;  public DetailAdapter(IDetailBusinessLogicContract detailBusinessLogicContract, ILogger logger)  {  \_detailBusinessLogicContract = detailBusinessLogicContract;  \_logger = logger;  var config = new MapperConfiguration(cfg =>  {  cfg.CreateMap<DetailBindingModel, DetailDataModel>();  cfg.CreateMap<DetailDataModel, DetailViewModel>();  });  \_mapper = new Mapper(config);  }  public DetailOperationResponse GetList()  {  try  {  return DetailOperationResponse.OK([  ..\_detailBusinessLogicContract.GetAllDetails()  .Select(x => \_mapper.Map<DetailDataModel, DetailViewModel>(x))  ]);  }  catch (NullListException)  {  \_logger.LogError("NullListException");  return DetailOperationResponse.NotFound("The list is not initialized");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return DetailOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return DetailOperationResponse.InternalServerError(ex.Message);  }  }  public DetailOperationResponse GetElementsByCreationDate(DateTime? from = null, DateTime? to = null)  {  try  {  return DetailOperationResponse.OK([  ..\_detailBusinessLogicContract.GetDetailsByCreationDate(from, to)  .Select(x => \_mapper.Map<DetailDataModel, DetailViewModel>(x))  ]);  }  catch (NullListException)  {  \_logger.LogError("NullListException");  return DetailOperationResponse.NotFound("The list is not initialized");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return DetailOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return DetailOperationResponse.InternalServerError(ex.Message);  }  }  public DetailOperationResponse GetElement(string data)  {  try  {  return DetailOperationResponse.OK(  \_mapper.Map<DetailViewModel>(\_detailBusinessLogicContract.GetDetailByData(data)));  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return DetailOperationResponse.BadRequest("Data is empty");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return DetailOperationResponse.NotFound($"Not found element by data {data}");  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return DetailOperationResponse.BadRequest($"Element by data: {data} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return DetailOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return DetailOperationResponse.InternalServerError(ex.Message);  }  }  public DetailOperationResponse CreateDetail(DetailBindingModel detailModel)  {  try  {  detailModel.Id = Guid.NewGuid().ToString();  \_detailBusinessLogicContract.InsertDetail(\_mapper.Map<DetailDataModel>(detailModel));  return DetailOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return DetailOperationResponse.BadRequest("Data is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return DetailOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementExistsException ex)  {  \_logger.LogError(ex, "ElementExistsException");  return DetailOperationResponse.BadRequest(ex.Message);  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return DetailOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return DetailOperationResponse.InternalServerError(ex.Message);  }  }  public DetailOperationResponse UpdateDetail(DetailBindingModel detailModel)  {  try  {  \_detailBusinessLogicContract.UpdateDetail(\_mapper.Map<DetailDataModel>(detailModel));  return DetailOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return DetailOperationResponse.BadRequest("Data is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return DetailOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return DetailOperationResponse.BadRequest($"Not found element by Id {detailModel.Id}");  }  catch (ElementExistsException ex)  {  \_logger.LogError(ex, "ElementExistsException");  return DetailOperationResponse.BadRequest(ex.Message);  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return DetailOperationResponse.BadRequest($"Element by id: {detailModel.Id} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return DetailOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return DetailOperationResponse.InternalServerError(ex.Message);  }  }  public DetailOperationResponse DeleteDetail(string id)  {  try  {  \_detailBusinessLogicContract.DeleteDetail(id);  return DetailOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return DetailOperationResponse.BadRequest("Id is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return DetailOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return DetailOperationResponse.BadRequest($"Not found element by id: {id}");  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return DetailOperationResponse.BadRequest($"Element by id: {id} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return DetailOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return DetailOperationResponse.InternalServerError(ex.Message);  }  }  }  // ReportAdapter.cs  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.ViewModels;  namespace GoToWorkApi.Adapters;  public class ReportAdapter : IReportAdapter  {  private readonly IReportContract \_reportBusinessLogic;  private readonly ILogger \_logger;  public ReportAdapter(IReportContract reportBusinessLogic, ILogger<ReportAdapter> logger)  {  \_reportBusinessLogic = reportBusinessLogic;  \_logger = logger;  }  public async Task<ReportOperationResponse> GetWorkshopsByDetailsAsync(WorkshopsReportBindingModel selectedDetailIds,  CancellationToken ct)  {  try  {  var data = await \_reportBusinessLogic.GetWorkshopsByDetailsAsync(selectedDetailIds.DetailIds, ct);  return ReportOperationResponse.OK(data);  }  catch (Exception ex)  {  \_logger.LogError(ex, "Error getting workshops by details report");  return ReportOperationResponse.InternalServerError(ex.Message);  }  }  public async Task<ReportOperationResponse> GetDetailsByMachinesAndProductionsAsync(  DetailsReportBindingModel selectedDates, CancellationToken ct)  {  try  {  var data = await \_reportBusinessLogic.GetDetailsByMachinesAndProductionsAsync(selectedDates.startDate,  selectedDates.endDate, ct);  return ReportOperationResponse.OK(data);  }  catch (Exception ex)  {  \_logger.LogError(ex, "Error getting details by machines and productions report");  return ReportOperationResponse.InternalServerError(ex.Message);  }  }  public async Task<ReportOperationResponse> CreateDocxDocumentWorkshopsByDetailsAsync(  WorkshopsReportBindingModel selectedDetailIds, CancellationToken ct)  {  try  {  var stream =  await \_reportBusinessLogic.CreateDocxDocumentWorkshopsByDetailsAsync(  await \_reportBusinessLogic.GetWorkshopsByDetailsAsync(  selectedDetailIds.DetailIds, ct), ct);  return ReportOperationResponse.OK(stream, "WorkshopsReportViewModel.docx");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Error creating DOCX document");  return ReportOperationResponse.InternalServerError(ex.Message);  }  }  public async Task<ReportOperationResponse> CreateXlsxDocumentWorkshopsByDetailsAsync(  WorkshopsReportBindingModel selectedDetailIds, CancellationToken ct)  {  try  {  var stream =  await \_reportBusinessLogic.CreateXlsxDocumentWorkshopsByDetailsAsync(  await \_reportBusinessLogic.GetWorkshopsByDetailsAsync(  selectedDetailIds.DetailIds, ct), ct);  return ReportOperationResponse.OK(stream, "WorkshopsReportViewModel.xlsx");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Error creating XLSX document");  return ReportOperationResponse.InternalServerError(ex.Message);  }  }  public async Task<ReportOperationResponse> CreatePdfDocumentDetailsByMachinesAndProductionsAsync(  DetailsReportBindingModel selectedDates, CancellationToken ct)  {  try  {  var stream =  await \_reportBusinessLogic.CreatePdfDocumentDetailsByMachinesAndProductionsAsync(  await \_reportBusinessLogic.GetDetailsByMachinesAndProductionsAsync(  selectedDates.startDate, selectedDates.endDate, ct),  selectedDates.startDate, selectedDates.endDate, ct);  return ReportOperationResponse.OK(stream, "DetailsReport.pdf");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Error creating PDF document");  return ReportOperationResponse.InternalServerError(ex.Message);  }  }  public async Task<ReportOperationResponse> SendPdfDocumentDetailsByMachinesAndProductionsEmailAsync(  DetailsReportBindingModel selectedDates, CancellationToken ct)  {  try  {  var report = await \_reportBusinessLogic.CreatePdfDocumentDetailsByMachinesAndProductionsAsync(  await \_reportBusinessLogic.GetDetailsByMachinesAndProductionsAsync(  selectedDates.startDate, selectedDates.endDate, ct),  selectedDates.startDate, selectedDates.endDate, ct);  await \_reportBusinessLogic.SendEmailAsync(report, selectedDates.email!, "Отчёт о деталях",  "DetailsReport.pdf", "application/pdf");  return ReportOperationResponse.NoContent();  }  catch (Exception ex)  {  \_logger.LogError(ex, "Error sending email");  return ReportOperationResponse.InternalServerError(ex.Message);  }  }  }  // ProductAdapter.cs  using AutoMapper;  using GoToWorkContracts.AdapterContracts;  using GoToWorkContracts.AdapterContracts.OperationResponses;  using GoToWorkContracts.BindingModels;  using GoToWorkContracts.BusinessLogicContracts;  using GoToWorkContracts.DataModels;  using GoToWorkContracts.Exceptions;  using GoToWorkContracts.ViewModels;  namespace GoToWorkApi.Adapters;  public class ProductAdapter : IProductAdapter  {  private readonly ILogger<ProductAdapter> \_logger;  private readonly Mapper \_mapper;  private readonly IProductBusinessLogicContract \_productBusinessLogic;  public ProductAdapter(IProductBusinessLogicContract productBusinessLogic, ILogger<ProductAdapter> logger)  {  \_productBusinessLogic = productBusinessLogic;  \_logger = logger;  var config = new MapperConfiguration(cfg =>  {  cfg.CreateMap<ProductBindingModel, ProductDataModel>();  cfg.CreateMap<ProductDataModel, ProductViewModel>();  cfg.CreateMap<MachineBindingModel, MachineDataModel>();  cfg.CreateMap<MachineDataModel, MachineViewModel>();  cfg.CreateMap<DetailProductBindingModel, DetailProductDataModel>();  cfg.CreateMap<DetailProductDataModel, DetailProductViewModel>();  });  \_mapper = new Mapper(config);  }  public ProductOperationResponse GetList()  {  try  {  return ProductOperationResponse.OK([  ..\_productBusinessLogic.GetAllProducts()  .Select(x => \_mapper.Map<ProductDataModel, ProductViewModel>(x))  ]);  }  catch (NullListException)  {  \_logger.LogError("NullListException");  return ProductOperationResponse.NotFound("The list is not initialized");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return ProductOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return ProductOperationResponse.InternalServerError(ex.Message);  }  }  public ProductOperationResponse GetListByMachine(string machineId)  {  try  {  return ProductOperationResponse.OK([  ..\_productBusinessLogic.GetProductsByMachine(machineId)  .Select(x => \_mapper.Map<ProductDataModel, ProductViewModel>(x))  ]);  }  catch (NullListException)  {  \_logger.LogError("NullListException");  return ProductOperationResponse.NotFound("The list is not initialized");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return ProductOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return ProductOperationResponse.InternalServerError(ex.Message);  }  }  public ProductOperationResponse GetListByCreationDate(DateTime from, DateTime to)  {  try  {  return ProductOperationResponse.OK([  ..\_productBusinessLogic.GetProductsByCreationDate(from, to)  .Select(x => \_mapper.Map<ProductDataModel, ProductViewModel>(x))  ]);  }  catch (NullListException)  {  \_logger.LogError("NullListException");  return ProductOperationResponse.NotFound("The list is not initialized");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return ProductOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return ProductOperationResponse.InternalServerError(ex.Message);  }  }  public ProductOperationResponse GetElement(string data)  {  try  {  return ProductOperationResponse.OK(  \_mapper.Map<ProductViewModel>(\_productBusinessLogic.GetProductByData(data)));  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return ProductOperationResponse.BadRequest("Data is empty");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return ProductOperationResponse.NotFound($"Not found element by data {data}");  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return ProductOperationResponse.BadRequest($"Element by data: {data} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return ProductOperationResponse.InternalServerError(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return ProductOperationResponse.InternalServerError(ex.Message);  }  }  public ProductOperationResponse CreateProduct(ProductBindingModel productModel)  {  try  {  productModel.Id = Guid.NewGuid().ToString();  \_productBusinessLogic.InsertProduct(\_mapper.Map<ProductDataModel>(productModel));  return ProductOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return ProductOperationResponse.BadRequest("Data is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return ProductOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementExistsException ex)  {  \_logger.LogError(ex, "ElementExistsException");  return ProductOperationResponse.BadRequest(ex.Message);  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return ProductOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return ProductOperationResponse.InternalServerError(ex.Message);  }  }  public ProductOperationResponse UpdateProduct(ProductBindingModel productModel)  {  try  {  \_productBusinessLogic.UpdateProduct(\_mapper.Map<ProductDataModel>(productModel));  return ProductOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return ProductOperationResponse.BadRequest("Data is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return ProductOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return ProductOperationResponse.BadRequest($"Not found element by Id {productModel.Id}");  }  catch (ElementExistsException ex)  {  \_logger.LogError(ex, "ElementExistsException");  return ProductOperationResponse.BadRequest(ex.Message);  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return ProductOperationResponse.BadRequest($"Element by id: {productModel.Id} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return ProductOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return ProductOperationResponse.InternalServerError(ex.Message);  }  }  public ProductOperationResponse DeleteProduct(string id)  {  try  {  \_productBusinessLogic.DeleteProduct(id);  return ProductOperationResponse.NoContent();  }  catch (ArgumentNullException ex)  {  \_logger.LogError(ex, "ArgumentNullException");  return ProductOperationResponse.BadRequest("Id is empty");  }  catch (ValidationException ex)  {  \_logger.LogError(ex, "ValidationException");  return ProductOperationResponse.BadRequest($"Incorrect data transmitted: {ex.Message}");  }  catch (ElementNotFoundException ex)  {  \_logger.LogError(ex, "ElementNotFoundException");  return ProductOperationResponse.BadRequest($"Not found element by id: {id}");  }  catch (ElementDeletedException ex)  {  \_logger.LogError(ex, "ElementDeletedException");  return ProductOperationResponse.BadRequest($"Element by id: {id} was deleted");  }  catch (StorageException ex)  {  \_logger.LogError(ex, "StorageException");  return ProductOperationResponse.BadRequest(  $"Error while working with data storage: {ex.InnerException!.Message}");  }  catch (Exception ex)  {  \_logger.LogError(ex, "Exception");  return ProductOperationResponse.InternalServerError(ex.Message);  }  }  }  // GoToWorkApi.csproj.nuget.g.targets  ﻿<?xml version="1.0" encoding="utf-8" standalone="no"?>  <Project ToolsVersion="14.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">  <ImportGroup Condition=" '$(ExcludeRestorePackageImports)' != 'true' ">  <Import Project="$(NuGetPackageRoot)system.text.json/9.0.7/buildTransitive/net8.0/System.Text.Json.targets" Condition="Exists('$(NuGetPackageRoot)system.text.json/9.0.7/buildTransitive/net8.0/System.Text.Json.targets')" />  <Import Project="$(NuGetPackageRoot)microsoft.extensions.configuration.binder/9.0.0/buildTransitive/netstandard2.0/Microsoft.Extensions.Configuration.Binder.targets" Condition="Exists('$(NuGetPackageRoot)microsoft.extensions.configuration.binder/9.0.0/buildTransitive/netstandard2.0/Microsoft.Extensions.Configuration.Binder.targets')" />  <Import Project="$(NuGetPackageRoot)microsoft.extensions.options/9.0.7/buildTransitive/net8.0/Microsoft.Extensions.Options.targets" Condition="Exists('$(NuGetPackageRoot)microsoft.extensions.options/9.0.7/buildTransitive/net8.0/Microsoft.Extensions.Options.targets')" />  <Import Project="$(NuGetPackageRoot)microsoft.extensions.logging.abstractions/9.0.7/buildTransitive/net8.0/Microsoft.Extensions.Logging.Abstractions.targets" Condition="Exists('$(NuGetPackageRoot)microsoft.extensions.logging.abstractions/9.0.7/buildTransitive/net8.0/Microsoft.Extensions.Logging.Abstractions.targets')" />  <Import Project="$(NuGetPackageRoot)mono.texttemplating/3.0.0/buildTransitive/Mono.TextTemplating.targets" Condition="Exists('$(NuGetPackageRoot)mono.texttemplating/3.0.0/buildTransitive/Mono.TextTemplating.targets')" />  <Import Project="$(NuGetPackageRoot)microsoft.codeanalysis.analyzers/3.3.4/buildTransitive/Microsoft.CodeAnalysis.Analyzers.targets" Condition="Exists('$(NuGetPackageRoot)microsoft.codeanalysis.analyzers/3.3.4/buildTransitive/Microsoft.CodeAnalysis.Analyzers.targets')" />  </ImportGroup>  </Project>  // GoToWorkApi.AssemblyInfo.cs  //------------------------------------------------------------------------------  // <auto-generated>  // This code was generated by a tool.  //  // Changes to this file may cause incorrect behavior and will be lost if  // the code is regenerated.  // </auto-generated>  //------------------------------------------------------------------------------  using System;  using System.Reflection;  [assembly: System.Reflection.AssemblyCompanyAttribute("GoToWorkApi")]  [assembly: System.Reflection.AssemblyConfigurationAttribute("Release")]  [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]  [assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+8753cd23468d9b41857b154ffbb5889134ac4e4c")]  [assembly: System.Reflection.AssemblyProductAttribute("GoToWorkApi")]  [assembly: System.Reflection.AssemblyTitleAttribute("GoToWorkApi")]  [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]  // Generated by the MSBuild WriteCodeFragment class.  // GoToWorkApi.GlobalUsings.g.cs  // <auto-generated/>  global using global::Microsoft.AspNetCore.Builder;  global using global::Microsoft.AspNetCore.Hosting;  global using global::Microsoft.AspNetCore.Http;  global using global::Microsoft.AspNetCore.Routing;  global using global::Microsoft.Extensions.Configuration;  global using global::Microsoft.Extensions.DependencyInjection;  global using global::Microsoft.Extensions.Hosting;  global using global::Microsoft.Extensions.Logging;  global using global::System;  global using global::System.Collections.Generic;  global using global::System.IO;  global using global::System.Linq;  global using global::System.Net.Http;  global using global::System.Net.Http.Json;  global using global::System.Threading;  global using global::System.Threading.Tasks;  // GoToWorkApi.MvcApplicationPartsAssemblyInfo.cs  //------------------------------------------------------------------------------  // <auto-generated>  // This code was generated by a tool.  //  // Changes to this file may cause incorrect behavior and will be lost if  // the code is regenerated.  // </auto-generated>  //------------------------------------------------------------------------------  using System;  using System.Reflection;  [assembly: Microsoft.AspNetCore.Mvc.ApplicationParts.ApplicationPartAttribute("GoToWorkBusinessLogic")]  [assembly: Microsoft.AspNetCore.Mvc.ApplicationParts.ApplicationPartAttribute("GoToWorkContracts")]  [assembly: Microsoft.AspNetCore.Mvc.ApplicationParts.ApplicationPartAttribute("GoToWorkDatabase")]  [assembly: Microsoft.AspNetCore.Mvc.ApplicationParts.ApplicationPartAttribute("Microsoft.AspNetCore.OpenApi")]  // Generated by the MSBuild WriteCodeFragment class.  // .NETCoreApp,Version=v9.0.AssemblyAttributes.cs  // <autogenerated />  using System;  using System.Reflection;  [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v9.0", FrameworkDisplayName = ".NET 9.0")]  // GoToWorkApi.AssemblyInfo.cs  //------------------------------------------------------------------------------  // <auto-generated>  // This code was generated by a tool.  //  // Changes to this file may cause incorrect behavior and will be lost if  // the code is regenerated.  // </auto-generated>  //------------------------------------------------------------------------------  using System;  using System.Reflection;  [assembly: System.Reflection.AssemblyCompanyAttribute("GoToWorkApi")]  [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]  [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]  [assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+8753cd23468d9b41857b154ffbb5889134ac4e4c")]  [assembly: System.Reflection.AssemblyProductAttribute("GoToWorkApi")]  [assembly: System.Reflection.AssemblyTitleAttribute("GoToWorkApi")]  [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]  // Generated by the MSBuild WriteCodeFragment class.  // GoToWorkApi.GlobalUsings.g.cs  // <auto-generated/>  global using global::Microsoft.AspNetCore.Builder;  global using global::Microsoft.AspNetCore.Hosting;  global using global::Microsoft.AspNetCore.Http;  global using global::Microsoft.AspNetCore.Routing;  global using global::Microsoft.Extensions.Configuration;  global using global::Microsoft.Extensions.DependencyInjection;  global using global::Microsoft.Extensions.Hosting;  global using global::Microsoft.Extensions.Logging;  global using global::System;  global using global::System.Collections.Generic;  global using global::System.IO;  global using global::System.Linq;  global using global::System.Net.Http;  global using global::System.Net.Http.Json;  global using global::System.Threading;  global using global::System.Threading.Tasks;  // GoToWorkApi.MvcApplicationPartsAssemblyInfo.cs  //------------------------------------------------------------------------------  // <auto-generated>  // This code was generated by a tool.  //  // Changes to this file may cause incorrect behavior and will be lost if  // the code is regenerated.  // </auto-generated>  //------------------------------------------------------------------------------  using System;  using System.Reflection;  [assembly: Microsoft.AspNetCore.Mvc.ApplicationParts.ApplicationPartAttribute("GoToWorkBusinessLogic")]  [assembly: Microsoft.AspNetCore.Mvc.ApplicationParts.ApplicationPartAttribute("GoToWorkContracts")]  [assembly: Microsoft.AspNetCore.Mvc.ApplicationParts.ApplicationPartAttribute("GoToWorkDatabase")]  [assembly: Microsoft.AspNetCore.Mvc.ApplicationParts.ApplicationPartAttribute("Microsoft.AspNetCore.OpenApi")]  // Generated by the MSBuild WriteCodeFragment class.  // .NETCoreApp,Version=v9.0.AssemblyAttributes.cs  // <autogenerated />  using System;  using System.Reflection;  [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v9.0", FrameworkDisplayName = ".NET 9.0")]  // ConfigurationDatabase.cs  using GoToWorkContracts.Infrastructure;  namespace GoToWorkApi.Infrastructure;  public class ConfigurationDatabase(IConfiguration configuration) : IConfigurationDatabase  {  private readonly Lazy<DatabaseSettings> \_settings = new(() =>  configuration.GetValue<DatabaseSettings>("DatabaseSettings")  ?? throw new InvalidDataException(nameof(DatabaseSettings)));  public string ConnectionString => "Host=127.0.0.1;Port=5432;Database=GoToWork;Username=postgres;Password=postgres;";  }  // DatabaseSettings.cs  namespace GoToWorkApi.Infrastructure;  public class DatabaseSettings  {  public required string ConnectionString { get; set; }  }  // vite.config.ts  import { defineConfig } from 'vite'  import react from '@vitejs/plugin-react-swc'  // https://vite.dev/config/  export default defineConfig({  plugins: [react()],  server: {  proxy: {  '/api': {  target: 'https://localhost:7156',  changeOrigin: true,  secure: false,  }  }  }  })  // index.d.ts  declare namespace pLocate {  interface Options {  /\*\*  Number of concurrently pending promises returned by `tester`. Minimum: `1`.  @default Infinity  \*/  readonly concurrency?: number;  /\*\*  Preserve `input` order when searching.  Disable this to improve performance if you don't care about the order.  @default true  \*/  readonly preserveOrder?: boolean;  }  }  /\*\*  Get the first fulfilled promise that satisfies the provided testing function.  @param input - An iterable of promises/values to test.  @param tester - This function will receive resolved values from `input` and is expected to return a `Promise<boolean>` or `boolean`.  @returns A `Promise` that is fulfilled when `tester` resolves to `true` or the iterable is done, or rejects if any of the promises reject. The fulfilled value is the current iterable value or `undefined` if `tester` never resolved to `true`.  @example  ```  import pathExists = require('path-exists');  import pLocate = require('p-locate');  const files = [  'unicorn.png',  'rainbow.png', // Only this one actually exists on disk  'pony.png'  ];  (async () => {  const foundPath = await pLocate(files, file => pathExists(file));  console.log(foundPath);  //=> 'rainbow'  })();  ```  \*/  declare function pLocate<ValueType>(  input: Iterable<PromiseLike<ValueType> | ValueType>,  tester: (element: ValueType) => PromiseLike<boolean> | boolean,  options?: pLocate.Options  ): Promise<ValueType | undefined>;  export = pLocate;  // main.d.ts  export type Platform = 'browser' | 'node' | 'neutral'  export type Format = 'iife' | 'cjs' | 'esm'  export type Loader = 'base64' | 'binary' | 'copy' | 'css' | 'dataurl' | 'default' | 'empty' | 'file' | 'js' | 'json' | 'jsx' | 'local-css' | 'text' | 'ts' | 'tsx'  export type LogLevel = 'verbose' | 'debug' | 'info' | 'warning' | 'error' | 'silent'  export type Charset = 'ascii' | 'utf8'  export type Drop = 'console' | 'debugger'  export type AbsPaths = 'code' | 'log' | 'metafile'  interface CommonOptions {  /\*\* Documentation: https://esbuild.github.io/api/#sourcemap \*/  sourcemap?: boolean | 'linked' | 'inline' | 'external' | 'both'  /\*\* Documentation: https://esbuild.github.io/api/#legal-comments \*/  legalComments?: 'none' | 'inline' | 'eof' | 'linked' | 'external'  /\*\* Documentation: https://esbuild.github.io/api/#source-root \*/  sourceRoot?: string  /\*\* Documentation: https://esbuild.github.io/api/#sources-content \*/  sourcesContent?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#format \*/  format?: Format  /\*\* Documentation: https://esbuild.github.io/api/#global-name \*/  globalName?: string  /\*\* Documentation: https://esbuild.github.io/api/#target \*/  target?: string | string[]  /\*\* Documentation: https://esbuild.github.io/api/#supported \*/  supported?: Record<string, boolean>  /\*\* Documentation: https://esbuild.github.io/api/#platform \*/  platform?: Platform  /\*\* Documentation: https://esbuild.github.io/api/#mangle-props \*/  mangleProps?: RegExp  /\*\* Documentation: https://esbuild.github.io/api/#mangle-props \*/  reserveProps?: RegExp  /\*\* Documentation: https://esbuild.github.io/api/#mangle-props \*/  mangleQuoted?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#mangle-props \*/  mangleCache?: Record<string, string | false>  /\*\* Documentation: https://esbuild.github.io/api/#drop \*/  drop?: Drop[]  /\*\* Documentation: https://esbuild.github.io/api/#drop-labels \*/  dropLabels?: string[]  /\*\* Documentation: https://esbuild.github.io/api/#minify \*/  minify?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#minify \*/  minifyWhitespace?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#minify \*/  minifyIdentifiers?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#minify \*/  minifySyntax?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#line-limit \*/  lineLimit?: number  /\*\* Documentation: https://esbuild.github.io/api/#charset \*/  charset?: Charset  /\*\* Documentation: https://esbuild.github.io/api/#tree-shaking \*/  treeShaking?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#ignore-annotations \*/  ignoreAnnotations?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#jsx \*/  jsx?: 'transform' | 'preserve' | 'automatic'  /\*\* Documentation: https://esbuild.github.io/api/#jsx-factory \*/  jsxFactory?: string  /\*\* Documentation: https://esbuild.github.io/api/#jsx-fragment \*/  jsxFragment?: string  /\*\* Documentation: https://esbuild.github.io/api/#jsx-import-source \*/  jsxImportSource?: string  /\*\* Documentation: https://esbuild.github.io/api/#jsx-development \*/  jsxDev?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#jsx-side-effects \*/  jsxSideEffects?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#define \*/  define?: { [key: string]: string }  /\*\* Documentation: https://esbuild.github.io/api/#pure \*/  pure?: string[]  /\*\* Documentation: https://esbuild.github.io/api/#keep-names \*/  keepNames?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#abs-paths \*/  absPaths?: AbsPaths[]  /\*\* Documentation: https://esbuild.github.io/api/#color \*/  color?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#log-level \*/  logLevel?: LogLevel  /\*\* Documentation: https://esbuild.github.io/api/#log-limit \*/  logLimit?: number  /\*\* Documentation: https://esbuild.github.io/api/#log-override \*/  logOverride?: Record<string, LogLevel>  /\*\* Documentation: https://esbuild.github.io/api/#tsconfig-raw \*/  tsconfigRaw?: string | TsconfigRaw  }  export interface TsconfigRaw {  compilerOptions?: {  alwaysStrict?: boolean  baseUrl?: string  experimentalDecorators?: boolean  importsNotUsedAsValues?: 'remove' | 'preserve' | 'error'  jsx?: 'preserve' | 'react-native' | 'react' | 'react-jsx' | 'react-jsxdev'  jsxFactory?: string  jsxFragmentFactory?: string  jsxImportSource?: string  paths?: Record<string, string[]>  preserveValueImports?: boolean  strict?: boolean  target?: string  useDefineForClassFields?: boolean  verbatimModuleSyntax?: boolean  }  }  export interface BuildOptions extends CommonOptions {  /\*\* Documentation: https://esbuild.github.io/api/#bundle \*/  bundle?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#splitting \*/  splitting?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#preserve-symlinks \*/  preserveSymlinks?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#outfile \*/  outfile?: string  /\*\* Documentation: https://esbuild.github.io/api/#metafile \*/  metafile?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#outdir \*/  outdir?: string  /\*\* Documentation: https://esbuild.github.io/api/#outbase \*/  outbase?: string  /\*\* Documentation: https://esbuild.github.io/api/#external \*/  external?: string[]  /\*\* Documentation: https://esbuild.github.io/api/#packages \*/  packages?: 'bundle' | 'external'  /\*\* Documentation: https://esbuild.github.io/api/#alias \*/  alias?: Record<string, string>  /\*\* Documentation: https://esbuild.github.io/api/#loader \*/  loader?: { [ext: string]: Loader }  /\*\* Documentation: https://esbuild.github.io/api/#resolve-extensions \*/  resolveExtensions?: string[]  /\*\* Documentation: https://esbuild.github.io/api/#main-fields \*/  mainFields?: string[]  /\*\* Documentation: https://esbuild.github.io/api/#conditions \*/  conditions?: string[]  /\*\* Documentation: https://esbuild.github.io/api/#write \*/  write?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#allow-overwrite \*/  allowOverwrite?: boolean  /\*\* Documentation: https://esbuild.github.io/api/#tsconfig \*/  tsconfig?: string  /\*\* Documentation: https://esbuild.github.io/api/#out-extension \*/  outExtension?: { [ext: string]: string }  /\*\* Documentation: https://esbuild.github.io/api/#public-path \*/  publicPath?: string  /\*\* Documentation: https://esbuild.github.io/api/#entry-names \*/  entryNames?: string  /\*\* Documentation: https://esbuild.github.io/api/#chunk-names \*/  chunkNames?: string  /\*\* Documentation: https://esbuild.github.io/api/#asset-names \*/  assetNames?: string  /\*\* Documentation: https://esbuild.github.io/api/#inject \*/  inject?: string[]  /\*\* Documentation: https://esbuild.github.io/api/#banner \*/  banner?: { [type: string]: string }  /\*\* Documentation: https://esbuild.github.io/api/#footer \*/  footer?: { [type: string]: string }  /\*\* Documentation: https://esbuild.github.io/api/#entry-points \*/  entryPoints?: (string | { in: string, out: string })[] | Record<string, string>  /\*\* Documentation: https://esbuild.github.io/api/#stdin \*/  stdin?: StdinOptions  /\*\* Documentation: https://esbuild.github.io/plugins/ \*/  plugins?: Plugin[]  /\*\* Documentation: https://esbuild.github.io/api/#working-directory \*/  absWorkingDir?: string  /\*\* Documentation: https://esbuild.github.io/api/#node-paths \*/  nodePaths?: string[]; // The "NODE\_PATH" variable from Node.js  }  export interface StdinOptions {  contents: string | Uint8Array  resolveDir?: string  sourcefile?: string  loader?: Loader  }  export interface Message {  id: string  pluginName: string  text: string  location: Location | null  notes: Note[]  /\*\*  \* Optional user-specified data that is passed through unmodified. You can  \* use this to stash the original error, for example.  \*/  detail: any  }  export interface Note {  text: string  location: Location | null  }  export interface Location {  file: string  namespace: string  /\*\* 1-based \*/  line: number  /\*\* 0-based, in bytes \*/  column: number  /\*\* in bytes \*/  length: number  lineText: string  suggestion: string  }  export interface OutputFile {  path: string  contents: Uint8Array  hash: string  /\*\* "contents" as text (changes automatically with "contents") \*/  readonly text: string  }  export interface BuildResult<ProvidedOptions extends BuildOptions = BuildOptions> {  errors: Message[]  warnings: Message[]  /\*\* Only when "write: false" \*/  outputFiles: OutputFile[] | (ProvidedOptions['write'] extends false ? never : undefined)  /\*\* Only when "metafile: true" \*/  metafile: Metafile | (ProvidedOptions['metafile'] extends true ? never : undefined)  /\*\* Only when "mangleCache" is present \*/  mangleCache: Record<string, string | false> | (ProvidedOptions['mangleCache'] extends Object ? never : undefined)  }  export interface BuildFailure extends Error {  errors: Message[]  warnings: Message[]  }  /\*\* Documentation: https://esbuild.github.io/api/#serve-arguments \*/  export interface ServeOptions {  port?: number  host?: string  servedir?: string  keyfile?: string  certfile?: string  fallback?: string  cors?: CORSOptions  onRequest?: (args: ServeOnRequestArgs) => void  }  /\*\* Documentation: https://esbuild.github.io/api/#cors \*/  export interface CORSOptions {  origin?: string | string[]  }  export interface ServeOnRequestArgs {  remoteAddress: string  method: string  path: string  status: number  /\*\* The time to generate the response, not to send it \*/  timeInMS: number  }  /\*\* Documentation: https://esbuild.github.io/api/#serve-return-values \*/  export interface ServeResult {  port: number  hosts: string[]  }  export interface TransformOptions extends CommonOptions {  /\*\* Documentation: https://esbuild.github.io/api/#sourcefile \*/  sourcefile?: string  /\*\* Documentation: https://esbuild.github.io/api/#loader \*/  loader?: Loader  /\*\* Documentation: https://esbuild.github.io/api/#banner \*/  banner?: string  /\*\* Documentation: https://esbuild.github.io/api/#footer \*/  footer?: string  }  export interface TransformResult<ProvidedOptions extends TransformOptions = TransformOptions> {  code: string  map: string  warnings: Message[]  /\*\* Only when "mangleCache" is present \*/  mangleCache: Record<string, string | false> | (ProvidedOptions['mangleCache'] extends Object ? never : undefined)  /\*\* Only when "legalComments" is "external" \*/  legalComments: string | (ProvidedOptions['legalComments'] extends 'external' ? never : undefined)  }  export interface TransformFailure extends Error {  errors: Message[]  warnings: Message[]  }  export interface Plugin {  name: string  setup: (build: PluginBuild) => (void | Promise<void>)  }  export interface PluginBuild {  /\*\* Documentation: https://esbuild.github.io/plugins/#build-options \*/  initialOptions: BuildOptions  /\*\* Documentation: https://esbuild.github.io/plugins/#resolve \*/  resolve(path: string, options?: ResolveOptions): Promise<ResolveResult>  /\*\* Documentation: https://esbuild.github.io/plugins/#on-start \*/  onStart(callback: () =>  (OnStartResult | null | void | Promise<OnStartResult | null | void>)): void  /\*\* Documentation: https://esbuild.github.io/plugins/#on-end \*/  onEnd(callback: (result: BuildResult) =>  (OnEndResult | null | void | Promise<OnEndResult | null | void>)): void  /\*\* Documentation: https://esbuild.github.io/plugins/#on-resolve \*/  onResolve(options: OnResolveOptions, callback: (args: OnResolveArgs) =>  (OnResolveResult | null | undefined | Promise<OnResolveResult | null | undefined>)): void  /\*\* Documentation: https://esbuild.github.io/plugins/#on-load \*/  onLoad(options: OnLoadOptions, callback: (args: OnLoadArgs) =>  (OnLoadResult | null | undefined | Promise<OnLoadResult | null | undefined>)): void  /\*\* Documentation: https://esbuild.github.io/plugins/#on-dispose \*/  onDispose(callback: () => void): void  // This is a full copy of the esbuild library in case you need it  esbuild: {  context: typeof context,  build: typeof build,  buildSync: typeof buildSync,  transform: typeof transform,  transformSync: typeof transformSync,  formatMessages: typeof formatMessages,  formatMessagesSync: typeof formatMessagesSync,  analyzeMetafile: typeof analyzeMetafile,  analyzeMetafileSync: typeof analyzeMetafileSync,  initialize: typeof initialize,  version: typeof version,  }  }  /\*\* Documentation: https://esbuild.github.io/plugins/#resolve-options \*/  export interface ResolveOptions {  pluginName?: string  importer?: string  namespace?: string  resolveDir?: string  kind?: ImportKind  pluginData?: any  with?: Record<string, string>  }  /\*\* Documentation: https://esbuild.github.io/plugins/#resolve-results \*/  export interface ResolveResult {  errors: Message[]  warnings: Message[]  path: string  external: boolean  sideEffects: boolean  namespace: string  suffix: string  pluginData: any  }  export interface OnStartResult {  errors?: PartialMessage[]  warnings?: PartialMessage[]  }  export interface OnEndResult {  errors?: PartialMessage[]  warnings?: PartialMessage[]  }  /\*\* Documentation: https://esbuild.github.io/plugins/#on-resolve-options \*/  export interface OnResolveOptions {  filter: RegExp  namespace?: string  }  /\*\* Documentation: https://esbuild.github.io/plugins/#on-resolve-arguments \*/  export interface OnResolveArgs {  path: string  importer: string  namespace: string  resolveDir: string  kind: ImportKind  pluginData: any  with: Record<string, string>  }  export type ImportKind =  | 'entry-point'  // JS  | 'import-statement'  | 'require-call'  | 'dynamic-import'  | 'require-resolve'  // CSS  | 'import-rule'  | 'composes-from'  | 'url-token'  /\*\* Documentation: https://esbuild.github.io/plugins/#on-resolve-results \*/  export interface OnResolveResult {  pluginName?: string  errors?: PartialMessage[]  warnings?: PartialMessage[]  path?: string  external?: boolean  sideEffects?: boolean  namespace?: string  suffix?: string  pluginData?: any  watchFiles?: string[]  watchDirs?: string[]  }  /\*\* Documentation: https://esbuild.github.io/plugins/#on-load-options \*/  export interface OnLoadOptions {  filter: RegExp  namespace?: string  }  /\*\* Documentation: https://esbuild.github.io/plugins/#on-load-arguments \*/  export interface OnLoadArgs {  path: string  namespace: string  suffix: string  pluginData: any  with: Record<string, string>  }  /\*\* Documentation: https://esbuild.github.io/plugins/#on-load-results \*/  export interface OnLoadResult {  pluginName?: string  errors?: PartialMessage[]  warnings?: PartialMessage[]  contents?: string | Uint8Array  resolveDir?: string  loader?: Loader  pluginData?: any  watchFiles?: string[]  watchDirs?: string[]  }  export interface PartialMessage {  id?: string  pluginName?: string  text?: string  location?: Partial<Location> | null  notes?: PartialNote[]  detail?: any  }  export interface PartialNote {  text?: string  location?: Partial<Location> | null  }  /\*\* Documentation: https://esbuild.github.io/api/#metafile \*/  export interface Metafile {  inputs: {  [path: string]: {  bytes: number  imports: {  path: string  kind: ImportKind  external?: boolean  original?: string  with?: Record<string, string>  }[]  format?: 'cjs' | 'esm'  with?: Record<string, string>  }  }  outputs: {  [path: string]: {  bytes: number  inputs: {  [path: string]: {  bytesInOutput: number  }  }  imports: {  path: string  kind: ImportKind | 'file-loader'  external?: boolean  }[]  exports: string[]  entryPoint?: string  cssBundle?: string  }  }  }  export interface FormatMessagesOptions {  kind: 'error' | 'warning'  color?: boolean  terminalWidth?: number  }  export interface AnalyzeMetafileOptions {  color?: boolean  verbose?: boolean  }  /\*\* Documentation: https://esbuild.github.io/api/#watch-arguments \*/  export interface WatchOptions {  delay?: number // In milliseconds  }  export interface BuildContext<ProvidedOptions extends BuildOptions = BuildOptions> {  /\*\* Documentation: https://esbuild.github.io/api/#rebuild \*/  rebuild(): Promise<BuildResult<ProvidedOptions>>  /\*\* Documentation: https://esbuild.github.io/api/#watch \*/  watch(options?: WatchOptions): Promise<void>  /\*\* Documentation: https://esbuild.github.io/api/#serve \*/  serve(options?: ServeOptions): Promise<ServeResult>  cancel(): Promise<void>  dispose(): Promise<void>  }  // This is a TypeScript type-level function which replaces any keys in "In"  // that aren't in "Out" with "never". We use this to reject properties with  // typos in object literals. See: https://stackoverflow.com/questions/49580725  type SameShape<Out, In extends Out> = In & { [Key in Exclude<keyof In, keyof Out>]: never }  /\*\*  \* This function invokes the "esbuild" command-line tool for you. It returns a  \* promise that either resolves with a "BuildResult" object or rejects with a  \* "BuildFailure" object.  \*  \* - Works in node: yes  \* - Works in browser: yes  \*  \* Documentation: https://esbuild.github.io/api/#build  \*/  export declare function build<T extends BuildOptions>(options: SameShape<BuildOptions, T>): Promise<BuildResult<T>>  /\*\*  \* This is the advanced long-running form of "build" that supports additional  \* features such as watch mode and a local development server.  \*  \* - Works in node: yes  \* - Works in browser: no  \*  \* Documentation: https://esbuild.github.io/api/#build  \*/  export declare function context<T extends BuildOptions>(options: SameShape<BuildOptions, T>): Promise<BuildContext<T>>  /\*\*  \* This function transforms a single JavaScript file. It can be used to minify  \* JavaScript, convert TypeScript/JSX to JavaScript, or convert newer JavaScript  \* to older JavaScript. It returns a promise that is either resolved with a  \* "TransformResult" object or rejected with a "TransformFailure" object.  \*  \* - Works in node: yes  \* - Works in browser: yes  \*  \* Documentation: https://esbuild.github.io/api/#transform  \*/  export declare function transform<T extends TransformOptions>(input: string | Uint8Array, options?: SameShape<TransformOptions, T>): Promise<TransformResult<T>>  /\*\*  \* Converts log messages to formatted message strings suitable for printing in  \* the terminal. This allows you to reuse the built-in behavior of esbuild's  \* log message formatter. This is a batch-oriented API for efficiency.  \*  \* - Works in node: yes  \* - Works in browser: yes  \*/  export declare function formatMessages(messages: PartialMessage[], options: FormatMessagesOptions): Promise<string[]>  /\*\*  \* Pretty-prints an analysis of the metafile JSON to a string. This is just for  \* convenience to be able to match esbuild's pretty-printing exactly. If you want  \* to customize it, you can just inspect the data in the metafile yourself.  \*  \* - Works in node: yes  \* - Works in browser: yes  \*  \* Documentation: https://esbuild.github.io/api/#analyze  \*/  export declare function analyzeMetafile(metafile: Metafile | string, options?: AnalyzeMetafileOptions): Promise<string>  /\*\*  \* A synchronous version of "build".  \*  \* - Works in node: yes  \* - Works in browser: no  \*  \* Documentation: https://esbuild.github.io/api/#build  \*/  export declare function buildSync<T extends BuildOptions>(options: SameShape<BuildOptions, T>): BuildResult<T>  /\*\*  \* A synchronous version of "transform".  \*  \* - Works in node: yes  \* - Works in browser: no  \*  \* Documentation: https://esbuild.github.io/api/#transform  \*/  export declare function transformSync<T extends TransformOptions>(input: string | Uint8Array, options?: SameShape<TransformOptions, T>): TransformResult<T>  /\*\*  \* A synchronous version of "formatMessages".  \*  \* - Works in node: yes  \* - Works in browser: no  \*/  export declare function formatMessagesSync(messages: PartialMessage[], options: FormatMessagesOptions): string[]  /\*\*  \* A synchronous version of "analyzeMetafile".  \*  \* - Works in node: yes  \* - Works in browser: no  \*  \* Documentation: https://esbuild.github.io/api/#analyze  \*/  export declare function analyzeMetafileSync(metafile: Metafile | string, options?: AnalyzeMetafileOptions): string  /\*\*  \* This configures the browser-based version of esbuild. It is necessary to  \* call this first and wait for the returned promise to be resolved before  \* making other API calls when using esbuild in the browser.  \*  \* - Works in node: yes  \* - Works in browser: yes ("options" is required)  \*  \* Documentation: https://esbuild.github.io/api/#browser  \*/  export declare function initialize(options: InitializeOptions): Promise<void>  export interface InitializeOptions {  /\*\*  \* The URL of the "esbuild.wasm" file. This must be provided when running  \* esbuild in the browser.  \*/  wasmURL?: string | URL  /\*\*  \* The result of calling "new WebAssembly.Module(buffer)" where "buffer"  \* is a typed array or ArrayBuffer containing the binary code of the  \* "esbuild.wasm" file.  \*  \* You can use this as an alternative to "wasmURL" for environments where it's  \* not possible to download the WebAssembly module.  \*/  wasmModule?: WebAssembly.Module  /\*\*  \* By default esbuild runs the WebAssembly-based browser API in a web worker  \* to avoid blocking the UI thread. This can be disabled by setting "worker"  \* to false.  \*/  worker?: boolean  }  export let version: string  // Call this function to terminate esbuild's child process. The child process  // is not terminated and re-created after each API call because it's more  // efficient to keep it around when there are multiple API calls.  //  // In node this happens automatically before the parent node process exits. So  // you only need to call this if you know you will not make any more esbuild  // API calls and you want to clean up resources.  //  // Unlike node, Deno lacks the necessary APIs to clean up child processes  // automatically. You must manually call stop() in Deno when you're done  // using esbuild or Deno will continue running forever.  //  // Another reason you might want to call this is if you are using esbuild from  // within a Deno test. Deno fails tests that create a child process without  // killing it before the test ends, so you have to call this function (and  // await the returned promise) in every Deno test that uses esbuild.  export declare function stop(): Promise<void>  // Note: These declarations exist to avoid type errors when you omit "dom" from  // "lib" in your "tsconfig.json" file. TypeScript confusingly declares the  // global "WebAssembly" type in "lib.dom.d.ts" even though it has nothing to do  // with the browser DOM and is present in many non-browser JavaScript runtimes  // (e.g. node and deno). Declaring it here allows esbuild's API to be used in  // these scenarios.  //  // There's an open issue about getting this problem corrected (although these  // declarations will need to remain even if this is fixed for backward  // compatibility with older TypeScript versions):  //  // https://github.com/microsoft/TypeScript-DOM-lib-generator/issues/826  //  declare global {  namespace WebAssembly {  interface Module {  }  }  interface URL {  }  }  // settings.d.ts  import \* as fs from './adapters/fs';  export interface Options {  followSymbolicLink?: boolean;  fs?: Partial<fs.FileSystemAdapter>;  markSymbolicLink?: boolean;  throwErrorOnBrokenSymbolicLink?: boolean;  }  export default class Settings {  private readonly \_options;  readonly followSymbolicLink: boolean;  readonly fs: fs.FileSystemAdapter;  readonly markSymbolicLink: boolean;  readonly throwErrorOnBrokenSymbolicLink: boolean;  constructor(\_options?: Options);  private \_getValue;  }  // index.d.ts  import type { FileSystemAdapter, StatAsynchronousMethod, StatSynchronousMethod } from './adapters/fs';  import \* as async from './providers/async';  import Settings, { Options } from './settings';  import type { Stats } from './types';  declare type AsyncCallback = async.AsyncCallback;  declare function stat(path: string, callback: AsyncCallback): void;  declare function stat(path: string, optionsOrSettings: Options | Settings, callback: AsyncCallback): void;  declare namespace stat {  function \_\_promisify\_\_(path: string, optionsOrSettings?: Options | Settings): Promise<Stats>;  }  declare function statSync(path: string, optionsOrSettings?: Options | Settings): Stats;  export { Settings, stat, statSync, AsyncCallback, FileSystemAdapter, StatAsynchronousMethod, StatSynchronousMethod, Options, Stats };  // async.d.ts  import type Settings from '../settings';  import type { ErrnoException, Stats } from '../types';  export declare type AsyncCallback = (error: ErrnoException, stats: Stats) => void;  export declare function read(path: string, settings: Settings, callback: AsyncCallback): void;  // sync.d.ts  import type Settings from '../settings';  import type { Stats } from '../types';  export declare function read(path: string, settings: Settings): Stats;  // fs.d.ts  /// <reference types="node" />  import \* as fs from 'fs';  import type { ErrnoException } from '../types';  export declare type StatAsynchronousMethod = (path: string, callback: (error: ErrnoException | null, stats: fs.Stats) => void) => void;  export declare type StatSynchronousMethod = (path: string) => fs.Stats;  export interface FileSystemAdapter {  lstat: StatAsynchronousMethod;  stat: StatAsynchronousMethod;  lstatSync: StatSynchronousMethod;  statSync: StatSynchronousMethod;  }  export declare const FILE\_SYSTEM\_ADAPTER: FileSystemAdapter;  export declare function createFileSystemAdapter(fsMethods?: Partial<FileSystemAdapter>): FileSystemAdapter;  // index.d.ts  /// <reference types="node" />  import type \* as fs from 'fs';  export declare type Stats = fs.Stats;  export declare type ErrnoException = NodeJS.ErrnoException;  // settings.d.ts  import \* as fsScandir from '@nodelib/fs.scandir';  import type { Entry, Errno } from './types';  export declare type FilterFunction<T> = (value: T) => boolean;  export declare type DeepFilterFunction = FilterFunction<Entry>;  export declare type EntryFilterFunction = FilterFunction<Entry>;  export declare type ErrorFilterFunction = FilterFunction<Errno>;  export interface Options {  basePath?: string;  concurrency?: number;  deepFilter?: DeepFilterFunction;  entryFilter?: EntryFilterFunction;  errorFilter?: ErrorFilterFunction;  followSymbolicLinks?: boolean;  fs?: Partial<fsScandir.FileSystemAdapter>;  pathSegmentSeparator?: string;  stats?: boolean;  throwErrorOnBrokenSymbolicLink?: boolean;  }  export default class Settings {  private readonly \_options;  readonly basePath?: string;  readonly concurrency: number;  readonly deepFilter: DeepFilterFunction | null;  readonly entryFilter: EntryFilterFunction | null;  readonly errorFilter: ErrorFilterFunction | null;  readonly pathSegmentSeparator: string;  readonly fsScandirSettings: fsScandir.Settings;  constructor(\_options?: Options);  private \_getValue;  }  // index.d.ts  /// <reference types="node" />  import type { Readable } from 'stream';  import type { Dirent, FileSystemAdapter } from '@nodelib/fs.scandir';  import { AsyncCallback } from './providers/async';  import Settings, { DeepFilterFunction, EntryFilterFunction, ErrorFilterFunction, Options } from './settings';  import type { Entry } from './types';  declare function walk(directory: string, callback: AsyncCallback): void;  declare function walk(directory: string, optionsOrSettings: Options | Settings, callback: AsyncCallback): void;  declare namespace walk {  function \_\_promisify\_\_(directory: string, optionsOrSettings?: Options | Settings): Promise<Entry[]>;  }  declare function walkSync(directory: string, optionsOrSettings?: Options | Settings): Entry[];  declare function walkStream(directory: string, optionsOrSettings?: Options | Settings): Readable;  export { walk, walkSync, walkStream, Settings, AsyncCallback, Dirent, Entry, FileSystemAdapter, Options, DeepFilterFunction, EntryFilterFunction, ErrorFilterFunction };  // async.d.ts  import AsyncReader from '../readers/async';  import type Settings from '../settings';  import type { Entry, Errno } from '../types';  export declare type AsyncCallback = (error: Errno, entries: Entry[]) => void;  export default class AsyncProvider {  private readonly \_root;  private readonly \_settings;  protected readonly \_reader: AsyncReader;  private readonly \_storage;  constructor(\_root: string, \_settings: Settings);  read(callback: AsyncCallback): void;  }  // sync.d.ts  import SyncReader from '../readers/sync';  import type Settings from '../settings';  import type { Entry } from '../types';  export default class SyncProvider {  private readonly \_root;  private readonly \_settings;  protected readonly \_reader: SyncReader;  constructor(\_root: string, \_settings: Settings);  read(): Entry[];  }  // index.d.ts  import AsyncProvider from './async';  import StreamProvider from './stream';  import SyncProvider from './sync';  export { AsyncProvider, StreamProvider, SyncProvider };  // stream.d.ts  /// <reference types="node" />  import { Readable } from 'stream';  import AsyncReader from '../readers/async';  import type Settings from '../settings';  export default class StreamProvider {  private readonly \_root;  private readonly \_settings;  protected readonly \_reader: AsyncReader;  protected readonly \_stream: Readable;  constructor(\_root: string, \_settings: Settings);  read(): Readable;  }  // async.d.ts  /// <reference types="node" />  import { EventEmitter } from 'events';  import \* as fsScandir from '@nodelib/fs.scandir';  import type Settings from '../settings';  import type { Entry, Errno } from '../types';  import Reader from './reader';  declare type EntryEventCallback = (entry: Entry) => void;  declare type ErrorEventCallback = (error: Errno) => void;  declare type EndEventCallback = () => void;  export default class AsyncReader extends Reader {  protected readonly \_settings: Settings;  protected readonly \_scandir: typeof fsScandir.scandir;  protected readonly \_emitter: EventEmitter;  private readonly \_queue;  private \_isFatalError;  private \_isDestroyed;  constructor(\_root: string, \_settings: Settings);  read(): EventEmitter;  get isDestroyed(): boolean;  destroy(): void;  onEntry(callback: EntryEventCallback): void;  onError(callback: ErrorEventCallback): void;  onEnd(callback: EndEventCallback): void;  private \_pushToQueue;  private \_worker;  private \_handleError;  private \_handleEntry;  private \_emitEntry;  }  export {};  // common.d.ts  import type { FilterFunction } from '../settings';  import type Settings from '../settings';  import type { Errno } from '../types';  export declare function isFatalError(settings: Settings, error: Errno): boolean;  export declare function isAppliedFilter<T>(filter: FilterFunction<T> | null, value: T): boolean;  export declare function replacePathSegmentSeparator(filepath: string, separator: string): string;  export declare function joinPathSegments(a: string, b: string, separator: string): string;  // sync.d.ts  import \* as fsScandir from '@nodelib/fs.scandir';  import type { Entry } from '../types';  import Reader from './reader';  export default class SyncReader extends Reader {  protected readonly \_scandir: typeof fsScandir.scandirSync;  private readonly \_storage;  private readonly \_queue;  read(): Entry[];  private \_pushToQueue;  private \_handleQueue;  private \_handleDirectory;  private \_handleError;  private \_handleEntry;  private \_pushToStorage;  }  // reader.d.ts  import type Settings from '../settings';  export default class Reader {  protected readonly \_root: string;  protected readonly \_settings: Settings;  constructor(\_root: string, \_settings: Settings);  }  // index.d.ts  /// <reference types="node" />  import type \* as scandir from '@nodelib/fs.scandir';  export declare type Entry = scandir.Entry;  export declare type Errno = NodeJS.ErrnoException;  export interface QueueItem {  directory: string;  base?: string;  }  // settings.d.ts  import \* as fsStat from '@nodelib/fs.stat';  import \* as fs from './adapters/fs';  export interface Options {  followSymbolicLinks?: boolean;  fs?: Partial<fs.FileSystemAdapter>;  pathSegmentSeparator?: string;  stats?: boolean;  throwErrorOnBrokenSymbolicLink?: boolean;  }  export default class Settings {  private readonly \_options;  readonly followSymbolicLinks: boolean;  readonly fs: fs.FileSystemAdapter;  readonly pathSegmentSeparator: string;  readonly stats: boolean;  readonly throwErrorOnBrokenSymbolicLink: boolean;  readonly fsStatSettings: fsStat.Settings;  constructor(\_options?: Options);  private \_getValue;  }  // index.d.ts  import type { FileSystemAdapter, ReaddirAsynchronousMethod, ReaddirSynchronousMethod } from './adapters/fs';  import \* as async from './providers/async';  import Settings, { Options } from './settings';  import type { Dirent, Entry } from './types';  declare type AsyncCallback = async.AsyncCallback;  declare function scandir(path: string, callback: AsyncCallback): void;  declare function scandir(path: string, optionsOrSettings: Options | Settings, callback: AsyncCallback): void;  declare namespace scandir {  function \_\_promisify\_\_(path: string, optionsOrSettings?: Options | Settings): Promise<Entry[]>;  }  declare function scandirSync(path: string, optionsOrSettings?: Options | Settings): Entry[];  export { scandir, scandirSync, Settings, AsyncCallback, Dirent, Entry, FileSystemAdapter, ReaddirAsynchronousMethod, ReaddirSynchronousMethod, Options };  // constants.d.ts  /\*\*  \* IS `true` for Node.js 10.10 and greater.  \*/  export declare const IS\_SUPPORT\_READDIR\_WITH\_FILE\_TYPES: boolean;  // fs.d.ts  import type { Dirent, Stats } from '../types';  export declare function createDirentFromStats(name: string, stats: Stats): Dirent;  // index.d.ts  import \* as fs from './fs';  export { fs };  // async.d.ts  /// <reference types="node" />  import type Settings from '../settings';  import type { Entry } from '../types';  export declare type AsyncCallback = (error: NodeJS.ErrnoException, entries: Entry[]) => void;  export declare function read(directory: string, settings: Settings, callback: AsyncCallback): void;  export declare function readdirWithFileTypes(directory: string, settings: Settings, callback: AsyncCallback): void;  export declare function readdir(directory: string, settings: Settings, callback: AsyncCallback): void;  // common.d.ts  export declare function joinPathSegments(a: string, b: string, separator: string): string;  // sync.d.ts  import type Settings from '../settings';  import type { Entry } from '../types';  export declare function read(directory: string, settings: Settings): Entry[];  export declare function readdirWithFileTypes(directory: string, settings: Settings): Entry[];  export declare function readdir(directory: string, settings: Settings): Entry[];  // fs.d.ts  import type \* as fsStat from '@nodelib/fs.stat';  import type { Dirent, ErrnoException } from '../types';  export interface ReaddirAsynchronousMethod {  (filepath: string, options: {  withFileTypes: true;  }, callback: (error: ErrnoException | null, files: Dirent[]) => void): void;  (filepath: string, callback: (error: ErrnoException | null, files: string[]) => void): void;  }  export interface ReaddirSynchronousMethod {  (filepath: string, options: {  withFileTypes: true;  }): Dirent[];  (filepath: string): string[];  }  export declare type FileSystemAdapter = fsStat.FileSystemAdapter & {  readdir: ReaddirAsynchronousMethod;  readdirSync: ReaddirSynchronousMethod;  };  export declare const FILE\_SYSTEM\_ADAPTER: FileSystemAdapter;  export declare function createFileSystemAdapter(fsMethods?: Partial<FileSystemAdapter>): FileSystemAdapter;  // index.d.ts  /// <reference types="node" />  import type \* as fs from 'fs';  export interface Entry {  dirent: Dirent;  name: string;  path: string;  stats?: Stats;  }  export declare type Stats = fs.Stats;  export declare type ErrnoException = NodeJS.ErrnoException;  export interface Dirent {  isBlockDevice: () => boolean;  isCharacterDevice: () => boolean;  isDirectory: () => boolean;  isFIFO: () => boolean;  isFile: () => boolean;  isSocket: () => boolean;  isSymbolicLink: () => boolean;  name: string;  }  // tslib.d.ts  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Copyright (c) Microsoft Corporation.  Permission to use, copy, modify, and/or distribute this software for any  purpose with or without fee is hereby granted.  THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH  REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY  AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,  INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM  LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR  OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR  PERFORMANCE OF THIS SOFTWARE.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/  /\*\*  \* Used to shim class extends.  \*  \* @param d The derived class.  \* @param b The base class.  \*/  export declare function \_\_extends(d: Function, b: Function): void;  /\*\*  \* Copy the values of all of the enumerable own properties from one or more source objects to a  \* target object. Returns the target object.  \*  \* @param t The target object to copy to.  \* @param sources One or more source objects from which to copy properties  \*/  export declare function \_\_assign(t: any, ...sources: any[]): any;  /\*\*  \* Performs a rest spread on an object.  \*  \* @param t The source value.  \* @param propertyNames The property names excluded from the rest spread.  \*/  export declare function \_\_rest(t: any, propertyNames: (string | symbol)[]): any;  /\*\*  \* Applies decorators to a target object  \*  \* @param decorators The set of decorators to apply.  \* @param target The target object.  \* @param key If specified, the own property to apply the decorators to.  \* @param desc The property descriptor, defaults to fetching the descriptor from the target object.  \* @experimental  \*/  export declare function \_\_decorate(decorators: Function[], target: any, key?: string | symbol, desc?: any): any;  /\*\*  \* Creates an observing function decorator from a parameter decorator.  \*  \* @param paramIndex The parameter index to apply the decorator to.  \* @param decorator The parameter decorator to apply. Note that the return value is ignored.  \* @experimental  \*/  export declare function \_\_param(paramIndex: number, decorator: Function): Function;  /\*\*  \* Applies decorators to a class or class member, following the native ECMAScript decorator specification.  \* @param ctor For non-field class members, the class constructor. Otherwise, `null`.  \* @param descriptorIn The `PropertyDescriptor` to use when unable to look up the property from `ctor`.  \* @param decorators The decorators to apply  \* @param contextIn The `DecoratorContext` to clone for each decorator application.  \* @param initializers An array of field initializer mutation functions into which new initializers are written.  \* @param extraInitializers An array of extra initializer functions into which new initializers are written.  \*/  export declare function \_\_esDecorate(ctor: Function | null, descriptorIn: object | null, decorators: Function[], contextIn: object, initializers: Function[] | null, extraInitializers: Function[]): void;  /\*\*  \* Runs field initializers or extra initializers generated by `\_\_esDecorate`.  \* @param thisArg The `this` argument to use.  \* @param initializers The array of initializers to evaluate.  \* @param value The initial value to pass to the initializers.  \*/  export declare function \_\_runInitializers(thisArg: unknown, initializers: Function[], value?: any): any;  /\*\*  \* Converts a computed property name into a `string` or `symbol` value.  \*/  export declare function \_\_propKey(x: any): string | symbol;  /\*\*  \* Assigns the name of a function derived from the left-hand side of an assignment.  \* @param f The function to rename.  \* @param name The new name for the function.  \* @param prefix A prefix (such as `"get"` or `"set"`) to insert before the name.  \*/  export declare function \_\_setFunctionName(f: Function, name: string | symbol, prefix?: string): Function;  /\*\*  \* Creates a decorator that sets metadata.  \*  \* @param metadataKey The metadata key  \* @param metadataValue The metadata value  \* @experimental  \*/  export declare function \_\_metadata(metadataKey: any, metadataValue: any): Function;  /\*\*  \* Converts a generator function into a pseudo-async function, by treating each `yield` as an `await`.  \*  \* @param thisArg The reference to use as the `this` value in the generator function  \* @param \_arguments The optional arguments array  \* @param P The optional promise constructor argument, defaults to the `Promise` property of the global object.  \* @param generator The generator function  \*/  export declare function \_\_awaiter(thisArg: any, \_arguments: any, P: Function, generator: Function): any;  /\*\*  \* Creates an Iterator object using the body as the implementation.  \*  \* @param thisArg The reference to use as the `this` value in the function  \* @param body The generator state-machine based implementation.  \*  \* @see [./docs/generator.md]  \*/  export declare function \_\_generator(thisArg: any, body: Function): any;  /\*\*  \* Creates bindings for all enumerable properties of `m` on `exports`  \*  \* @param m The source object  \* @param o The `exports` object.  \*/  export declare function \_\_exportStar(m: any, o: any): void;  /\*\*  \* Creates a value iterator from an `Iterable` or `ArrayLike` object.  \*  \* @param o The object.  \* @throws {TypeError} If `o` is neither `Iterable`, nor an `ArrayLike`.  \*/  export declare function \_\_values(o: any): any;  /\*\*  \* Reads values from an `Iterable` or `ArrayLike` object and returns the resulting array.  \*  \* @param o The object to read from.  \* @param n The maximum number of arguments to read, defaults to `Infinity`.  \*/  export declare function \_\_read(o: any, n?: number): any[];  /\*\*  \* Creates an array from iterable spread.  \*  \* @param args The Iterable objects to spread.  \* @deprecated since TypeScript 4.2 - Use `\_\_spreadArray`  \*/  export declare function \_\_spread(...args: any[][]): any[];  /\*\*  \* Creates an array from array spread.  \*  \* @param args The ArrayLikes to spread into the resulting array.  \* @deprecated since TypeScript 4.2 - Use `\_\_spreadArray`  \*/  export declare function \_\_spreadArrays(...args: any[][]): any[];  /\*\*  \* Spreads the `from` array into the `to` array.  \*  \* @param pack Replace empty elements with `undefined`.  \*/  export declare function \_\_spreadArray(to: any[], from: any[], pack?: boolean): any[];  /\*\*  \* Creates an object that signals to `\_\_asyncGenerator` that it shouldn't be yielded,  \* and instead should be awaited and the resulting value passed back to the generator.  \*  \* @param v The value to await.  \*/  export declare function \_\_await(v: any): any;  /\*\*  \* Converts a generator function into an async generator function, by using `yield \_\_await`  \* in place of normal `await`.  \*  \* @param thisArg The reference to use as the `this` value in the generator function  \* @param \_arguments The optional arguments array  \* @param generator The generator function  \*/  export declare function \_\_asyncGenerator(thisArg: any, \_arguments: any, generator: Function): any;  /\*\*  \* Used to wrap a potentially async iterator in such a way so that it wraps the result  \* of calling iterator methods of `o` in `\_\_await` instances, and then yields the awaited values.  \*  \* @param o The potentially async iterator.  \* @returns A synchronous iterator yielding `\_\_await` instances on every odd invocation  \* and returning the awaited `IteratorResult` passed to `next` every even invocation.  \*/  export declare function \_\_asyncDelegator(o: any): any;  /\*\*  \* Creates a value async iterator from an `AsyncIterable`, `Iterable` or `ArrayLike` object.  \*  \* @param o The object.  \* @throws {TypeError} If `o` is neither `AsyncIterable`, `Iterable`, nor an `ArrayLike`.  \*/  export declare function \_\_asyncValues(o: any): any;  /\*\*  \* Creates a `TemplateStringsArray` frozen object from the `cooked` and `raw` arrays.  \*  \* @param cooked The cooked possibly-sparse array.  \* @param raw The raw string content.  \*/  export declare function \_\_makeTemplateObject(cooked: string[], raw: string[]): TemplateStringsArray;  /\*\*  \* Used to shim default and named imports in ECMAScript Modules transpiled to CommonJS.  \*  \* ```js  \* import Default, { Named, Other } from "mod";  \* // or  \* import { default as Default, Named, Other } from "mod";  \* ```  \*  \* @param mod The CommonJS module exports object.  \*/  export declare function \_\_importStar<T>(mod: T): T;  /\*\*  \* Used to shim default imports in ECMAScript Modules transpiled to CommonJS.  \*  \* ```js  \* import Default from "mod";  \* ```  \*  \* @param mod The CommonJS module exports object.  \*/  export declare function \_\_importDefault<T>(mod: T): T | { default: T };  /\*\*  \* Emulates reading a private instance field.  \*  \* @param receiver The instance from which to read the private field.  \* @param state A WeakMap containing the private field value for an instance.  \* @param kind Either `"f"` for a field, `"a"` for an accessor, or `"m"` for a method.  \*  \* @throws {TypeError} If `state` doesn't have an entry for `receiver`.  \*/  export declare function \_\_classPrivateFieldGet<T extends object, V>(  receiver: T,  state: { has(o: T): boolean, get(o: T): V | undefined },  kind?: "f"  ): V;  /\*\*  \* Emulates reading a private static field.  \*  \* @param receiver The object from which to read the private static field.  \* @param state The class constructor containing the definition of the static field.  \* @param kind Either `"f"` for a field, `"a"` for an accessor, or `"m"` for a method.  \* @param f The descriptor that holds the static field value.  \*  \* @throws {TypeError} If `receiver` is not `state`.  \*/  export declare function \_\_classPrivateFieldGet<T extends new (...args: any[]) => unknown, V>(  receiver: T,  state: T,  kind: "f",  f: { value: V }  ): V;  /\*\*  \* Emulates evaluating a private instance "get" accessor.  \*  \* @param receiver The instance on which to evaluate the private "get" accessor.  \* @param state A WeakSet used to verify an instance supports the private "get" accessor.  \* @param kind Either `"f"` for a field, `"a"` for an accessor, or `"m"` for a method.  \* @param f The "get" accessor function to evaluate.  \*  \* @throws {TypeError} If `state` doesn't have an entry for `receiver`.  \*/  export declare function \_\_classPrivateFieldGet<T extends object, V>(  receiver: T,  state: { has(o: T): boolean },  kind: "a",  f: () => V  ): V;  /\*\*  \* Emulates evaluating a private static "get" accessor.  \*  \* @param receiver The object on which to evaluate the private static "get" accessor.  \* @param state The class constructor containing the definition of the static "get" accessor.  \* @param kind Either `"f"` for a field, `"a"` for an accessor, or `"m"` for a method.  \* @param f The "get" accessor function to evaluate.  \*  \* @throws {TypeError} If `receiver` is not `state`.  \*/  export declare function \_\_classPrivateFieldGet<T extends new (...args: any[]) => unknown, V>(  receiver: T,  state: T,  kind: "a",  f: () => V  ): V;  /\*\*  \* Emulates reading a private instance method.  \*  \* @param receiver The instance from which to read a private method.  \* @param state A WeakSet used to verify an instance supports the private method.  \* @param kind Either `"f"` for a field, `"a"` for an accessor, or `"m"` for a method.  \* @param f The function to return as the private instance method.  \*  \* @throws {TypeError} If `state` doesn't have an entry for `receiver`.  \*/  export declare function \_\_classPrivateFieldGet<T extends object, V extends (...args: any[]) => unknown>(  receiver: T,  state: { has(o: T): boolean },  kind: "m",  f: V  ): V;  /\*\*  \* Emulates reading a private static method.  \*  \* @param receiver The object from which to read the private static method.  \* @param state The class constructor containing the definition of the static method.  \* @param kind Either `"f"` for a field, `"a"` for an accessor, or `"m"` for a method.  \* @param f The function to return as the private static method.  \*  \* @throws {TypeError} If `receiver` is not `state`.  \*/  export declare function \_\_classPrivateFieldGet<T extends new (...args: any[]) => unknown, V extends (...args: any[]) => unknown>(  receiver: T,  state: T,  kind: "m",  f: V  ): V;  /\*\*  \* Emulates writing to a private instance field.  \*  \* @param receiver The instance on which to set a private field value.  \* @param state A WeakMap used to store the private field value for an instance.  \* @param value The value to store in the private field.  \* @param kind Either `"f"` for a field, `"a"` for an accessor, or `"m"` for a method.  \*  \* @throws {TypeError} If `state` doesn't have an entry for `receiver`.  \*/  export declare function \_\_classPrivateFieldSet<T extends object, V>(  receiver: T,  state: { has(o: T): boolean, set(o: T, value: V): unknown },  value: V,  kind?: "f"  ): V;  /\*\*  \* Emulates writing to a private static field.  \*  \* @param receiver The object on which to set the private static field.  \* @param state The class constructor containing the definition of the private static field.  \* @param value The value to store in the private field.  \* @param kind Either `"f"` for a field, `"a"` for an accessor, or `"m"` for a method.  \* @param f The descriptor that holds the static field value.  \*  \* @throws {TypeError} If `receiver` is not `state`.  \*/  export declare function \_\_classPrivateFieldSet<T extends new (...args: any[]) => unknown, V>(  receiver: T,  state: T,  value: V,  kind: "f",  f: { value: V }  ): V;  /\*\*  \* Emulates writing to a private instance "set" accessor.  \*  \* @param receiver The instance on which to evaluate the private instance "set" accessor.  \* @param state A WeakSet used to verify an instance supports the private "set" accessor.  \* @param value The value to store in the private accessor.  \* @param kind Either `"f"` for a field, `"a"` for an accessor, or `"m"` for a method.  \* @param f The "set" accessor function to evaluate.  \*  \* @throws {TypeError} If `state` doesn't have an entry for `receiver`.  \*/  export declare function \_\_classPrivateFieldSet<T extends object, V>(  receiver: T,  state: { has(o: T): boolean },  value: V,  kind: "a",  f: (v: V) => void  ): V;  /\*\*  \* Emulates writing to a private static "set" accessor.  \*  \* @param receiver The object on which to evaluate the private static "set" accessor.  \* @param state The class constructor containing the definition of the static "set" accessor.  \* @param value The value to store in the private field.  \* @param kind Either `"f"` for a field, `"a"` for an accessor, or `"m"` for a method.  \* @param f The "set" accessor function to evaluate.  \*  \* @throws {TypeError} If `receiver` is not `state`.  \*/  export declare function \_\_classPrivateFieldSet<T extends new (...args: any[]) => unknown, V>(  receiver: T,  state: T,  value: V,  kind: "a",  f: (v: V) => void  ): V;  /\*\*  \* Checks for the existence of a private field/method/accessor.  \*  \* @param state The class constructor containing the static member, or the WeakMap or WeakSet associated with a private instance member.  \* @param receiver The object for which to test the presence of the private member.  \*/  export declare function \_\_classPrivateFieldIn(  state: (new (...args: any[]) => unknown) | { has(o: any): boolean },  receiver: unknown,  ): boolean;  /\*\*  \* Creates a re-export binding on `object` with key `objectKey` that references `target[key]`.  \*  \* @param object The local `exports` object.  \* @param target The object to re-export from.  \* @param key The property key of `target` to re-export.  \* @param objectKey The property key to re-export as. Defaults to `key`.  \*/  export declare function \_\_createBinding(object: object, target: object, key: PropertyKey, objectKey?: PropertyKey): void;  /\*\*  \* Adds a disposable resource to a resource-tracking environment object.  \* @param env A resource-tracking environment object.  \* @param value Either a Disposable or AsyncDisposable object, `null`, or `undefined`.  \* @param async When `true`, `AsyncDisposable` resources can be added. When `false`, `AsyncDisposable` resources cannot be added.  \* @returns The {@link value} argument.  \*  \* @throws {TypeError} If {@link value} is not an object, or if either `Symbol.dispose` or `Symbol.asyncDispose` are not  \* defined, or if {@link value} does not have an appropriate `Symbol.dispose` or `Symbol.asyncDispose` method.  \*/  export declare function \_\_addDisposableResource<T>(env: { stack: { value?: unknown, dispose?: Function, async: boolean }[]; error: unknown; hasError: boolean; }, value: T, async: boolean): T;  /\*\*  \* Disposes all resources in a resource-tracking environment object.  \* @param env A resource-tracking environment object.  \* @returns A {@link Promise} if any resources in the environment were marked as `async` when added; otherwise, `void`.  \*  \* @throws {SuppressedError} if an error thrown during disposal would have suppressed a prior error from disposal or the  \* error recorded in the resource-tracking environment object.  \* @seealso {@link \_\_addDisposableResource}  \*/  export declare function \_\_disposeResources(env: { stack: { value?: unknown, dispose?: Function, async: boolean }[]; error: unknown; hasError: boolean; }): any;  /\*\*  \* Transforms a relative import specifier ending in a non-declaration TypeScript file extension to its JavaScript file extension counterpart.  \* @param path The import specifier.  \* @param preserveJsx Causes '\*.tsx' to transform to '\*.jsx' instead of '\*.js'. Should be true when `--jsx` is set to `preserve`.  \*/  export declare function \_\_rewriteRelativeImportExtension(path: string, preserveJsx?: boolean): string;  // index.d.ts  // Note: named reexports are used instead of `export \*` because  // TypeScript itself doesn't resolve the `export \*` when checking  // if a particular helper exists.  export {  \_\_extends,  \_\_assign,  \_\_rest,  \_\_decorate,  \_\_param,  \_\_esDecorate,  \_\_runInitializers,  \_\_propKey,  \_\_setFunctionName,  \_\_metadata,  \_\_awaiter,  \_\_generator,  \_\_exportStar,  \_\_values,  \_\_read,  \_\_spread,  \_\_spreadArrays,  \_\_spreadArray,  \_\_await,  \_\_asyncGenerator,  \_\_asyncDelegator,  \_\_asyncValues,  \_\_makeTemplateObject,  \_\_importStar,  \_\_importDefault,  \_\_classPrivateFieldGet,  \_\_classPrivateFieldSet,  \_\_classPrivateFieldIn,  \_\_createBinding,  \_\_addDisposableResource,  \_\_disposeResources,  \_\_rewriteRelativeImportExtension,  } from '../tslib.js';  export \* as default from '../tslib.js';  // index.d.cts  /\*\*  \* Generate secure URL-friendly unique ID.  \*  \* By default, the ID will have 21 symbols to have a collision probability  \* similar to UUID v4.  \*  \* ```js  \* import { nanoid } from 'nanoid'  \* model.id = nanoid() //=> "Uakgb\_J5m9g-0JDMbcJqL"  \* ```  \*  \* @param size Size of the ID. The default size is 21.  \* @returns A random string.  \*/  export function nanoid(size?: number): string  /\*\*  \* Generate secure unique ID with custom alphabet.  \*  \* Alphabet must contain 256 symbols or less. Otherwise, the generator  \* will not be secure.  \*  \* @param alphabet Alphabet used to generate the ID.  \* @param defaultSize Size of the ID. The default size is 21.  \* @returns A random string generator.  \*  \* ```js  \* const { customAlphabet } = require('nanoid')  \* const nanoid = customAlphabet('0123456789абвгдеё', 5)  \* nanoid() //=> "8ё56а"  \* ```  \*/  export function customAlphabet(  alphabet: string,  defaultSize?: number  ): (size?: number) => string  /\*\*  \* Generate unique ID with custom random generator and alphabet.  \*  \* Alphabet must contain 256 symbols or less. Otherwise, the generator  \* will not be secure.  \*  \* ```js  \* import { customRandom } from 'nanoid/format'  \*  \* const nanoid = customRandom('abcdef', 5, size => {  \* const random = []  \* for (let i = 0; i < size; i++) {  \* random.push(randomByte())  \* }  \* return random  \* })  \*  \* nanoid() //=> "fbaef"  \* ```  \*  \* @param alphabet Alphabet used to generate a random string.  \* @param size Size of the random string.  \* @param random A random bytes generator.  \* @returns A random string generator.  \*/  export function customRandom(  alphabet: string,  size: number,  random: (bytes: number) => Uint8Array  ): () => string  /\*\*  \* URL safe symbols.  \*  \* ```js  \* import { urlAlphabet } from 'nanoid'  \* const nanoid = customAlphabet(urlAlphabet, 10)  \* nanoid() //=> "Uakgb\_J5m9"  \* ```  \*/  export const urlAlphabet: string  /\*\*  \* Generate an array of random bytes collected from hardware noise.  \*  \* ```js  \* import { customRandom, random } from 'nanoid'  \* const nanoid = customRandom("abcdef", 5, random)  \* ```  \*  \* @param bytes Size of the array.  \* @returns An array of random bytes.  \*/  export function random(bytes: number): Uint8Array  // index.d.ts  /\*\*  \* Generate secure URL-friendly unique ID.  \*  \* By default, the ID will have 21 symbols to have a collision probability  \* similar to UUID v4.  \*  \* ```js  \* import { nanoid } from 'nanoid'  \* model.id = nanoid() //=> "Uakgb\_J5m9g-0JDMbcJqL"  \* ```  \*  \* @param size Size of the ID. The default size is 21.  \* @returns A random string.  \*/  export function nanoid(size?: number): string  /\*\*  \* Generate secure unique ID with custom alphabet.  \*  \* Alphabet must contain 256 symbols or less. Otherwise, the generator  \* will not be secure.  \*  \* @param alphabet Alphabet used to generate the ID.  \* @param defaultSize Size of the ID. The default size is 21.  \* @returns A random string generator.  \*  \* ```js  \* const { customAlphabet } = require('nanoid')  \* const nanoid = customAlphabet('0123456789абвгдеё', 5)  \* nanoid() //=> "8ё56а"  \* ```  \*/  export function customAlphabet(  alphabet: string,  defaultSize?: number  ): (size?: number) => string  /\*\*  \* Generate unique ID with custom random generator and alphabet.  \*  \* Alphabet must contain 256 symbols or less. Otherwise, the generator  \* will not be secure.  \*  \* ```js  \* import { customRandom } from 'nanoid/format'  \*  \* const nanoid = customRandom('abcdef', 5, size => {  \* const random = []  \* for (let i = 0; i < size; i++) {  \* random.push(randomByte())  \* }  \* return random  \* })  \*  \* nanoid() //=> "fbaef"  \* ```  \*  \* @param alphabet Alphabet used to generate a random string.  \* @param size Size of the random string.  \* @param random A random bytes generator.  \* @returns A random string generator.  \*/  export function customRandom(  alphabet: string,  size: number,  random: (bytes: number) => Uint8Array  ): () => string  /\*\*  \* URL safe symbols.  \*  \* ```js  \* import { urlAlphabet } from 'nanoid'  \* const nanoid = customAlphabet(urlAlphabet, 10)  \* nanoid() //=> "Uakgb\_J5m9"  \* ```  \*/  export const urlAlphabet: string  /\*\*  \* Generate an array of random bytes collected from hardware noise.  \*  \* ```js  \* import { customRandom, random } from 'nanoid'  \* const nanoid = customRandom("abcdef", 5, random)  \* ```  \*  \* @param bytes Size of the array.  \* @returns An array of random bytes.  \*/  export function random(bytes: number): Uint8Array  // index.d.ts  /\*\*  \* Generate secure URL-friendly unique ID. The non-blocking version.  \*  \* By default, the ID will have 21 symbols to have a collision probability  \* similar to UUID v4.  \*  \* ```js  \* import { nanoid } from 'nanoid/async'  \* nanoid().then(id => {  \* model.id = id  \* })  \* ```  \*  \* @param size Size of the ID. The default size is 21.  \* @returns A promise with a random string.  \*/  export function nanoid(size?: number): Promise<string>  /\*\*  \* A low-level function.  \* Generate secure unique ID with custom alphabet. The non-blocking version.  \*  \* Alphabet must contain 256 symbols or less. Otherwise, the generator  \* will not be secure.  \*  \* @param alphabet Alphabet used to generate the ID.  \* @param defaultSize Size of the ID. The default size is 21.  \* @returns A function that returns a promise with a random string.  \*  \* ```js  \* import { customAlphabet } from 'nanoid/async'  \* const nanoid = customAlphabet('0123456789абвгдеё', 5)  \* nanoid().then(id => {  \* model.id = id //=> "8ё56а"  \* })  \* ```  \*/  export function customAlphabet(  alphabet: string,  defaultSize?: number  ): (size?: number) => Promise<string>  /\*\*  \* Generate an array of random bytes collected from hardware noise.  \*  \* ```js  \* import { random } from 'nanoid/async'  \* random(5).then(bytes => {  \* bytes //=> [10, 67, 212, 67, 89]  \* })  \* ```  \*  \* @param bytes Size of the array.  \* @returns A promise with a random bytes array.  \*/  export function random(bytes: number): Promise<Uint8Array>  // index.d.ts  /\*\*  \* Generate URL-friendly unique ID. This method uses the non-secure  \* predictable random generator with bigger collision probability.  \*  \* ```js  \* import { nanoid } from 'nanoid/non-secure'  \* model.id = nanoid() //=> "Uakgb\_J5m9g-0JDMbcJqL"  \* ```  \*  \* @param size Size of the ID. The default size is 21.  \* @returns A random string.  \*/  export function nanoid(size?: number): string  /\*\*  \* Generate a unique ID based on a custom alphabet.  \* This method uses the non-secure predictable random generator  \* with bigger collision probability.  \*  \* @param alphabet Alphabet used to generate the ID.  \* @param defaultSize Size of the ID. The default size is 21.  \* @returns A random string generator.  \*  \* ```js  \* import { customAlphabet } from 'nanoid/non-secure'  \* const nanoid = customAlphabet('0123456789абвгдеё', 5)  \* model.id = //=> "8ё56а"  \* ```  \*/  export function customAlphabet(  alphabet: string,  defaultSize?: number  ): (size?: number) => string  // syntax.d.ts  declare const SyntaxError: SyntaxErrorConstructor;  export = SyntaxError;  // range.d.ts  declare const RangeError: RangeErrorConstructor;  export = RangeError;  // ref.d.ts  declare const ReferenceError: ReferenceErrorConstructor;  export = ReferenceError;  // eval.d.ts  declare const EvalError: EvalErrorConstructor;  export = EvalError;  // uri.d.ts  declare const URIError: URIErrorConstructor;  export = URIError;  // index.d.ts  declare const Error: ErrorConstructor;  export = Error;  // type.d.ts  declare const TypeError: TypeErrorConstructor  export = TypeError;  // client.d.ts  /// <reference path="./types/importMeta.d.ts" />  // CSS modules  type CSSModuleClasses = { readonly [key: string]: string }  declare module '\*.module.css' {  const classes: CSSModuleClasses  export default classes  }  declare module '\*.module.scss' {  const classes: CSSModuleClasses  export default classes  }  declare module '\*.module.sass' {  const classes: CSSModuleClasses  export default classes  }  declare module '\*.module.less' {  const classes: CSSModuleClasses  export default classes  }  declare module '\*.module.styl' {  const classes: CSSModuleClasses  export default classes  }  declare module '\*.module.stylus' {  const classes: CSSModuleClasses  export default classes  }  declare module '\*.module.pcss' {  const classes: CSSModuleClasses  export default classes  }  declare module '\*.module.sss' {  const classes: CSSModuleClasses  export default classes  }  // CSS  declare module '\*.css' {}  declare module '\*.scss' {}  declare module '\*.sass' {}  declare module '\*.less' {}  declare module '\*.styl' {}  declare module '\*.stylus' {}  declare module '\*.pcss' {}  declare module '\*.sss' {}  // Built-in asset types  // see `src/node/constants.ts`  // images  declare module '\*.apng' {  const src: string  export default src  }  declare module '\*.bmp' {  const src: string  export default src  }  declare module '\*.png' {  const src: string  export default src  }  declare module '\*.jpg' {  const src: string  export default src  }  declare module '\*.jpeg' {  const src: string  export default src  }  declare module '\*.jfif' {  const src: string  export default src  }  declare module '\*.pjpeg' {  const src: string  export default src  }  declare module '\*.pjp' {  const src: string  export default src  }  declare module '\*.gif' {  const src: string  export default src  }  declare module '\*.svg' {  const src: string  export default src  }  declare module '\*.ico' {  const src: string  export default src  }  declare module '\*.webp' {  const src: string  export default src  }  declare module '\*.avif' {  const src: string  export default src  }  declare module '\*.cur' {  const src: string  export default src  }  declare module '\*.jxl' {  const src: string  export default src  }  // media  declare module '\*.mp4' {  const src: string  export default src  }  declare module '\*.webm' {  const src: string  export default src  }  declare module '\*.ogg' {  const src: string  export default src  }  declare module '\*.mp3' {  const src: string  export default src  }  declare module '\*.wav' {  const src: string  export default src  }  declare module '\*.flac' {  const src: string  export default src  }  declare module '\*.aac' {  const src: string  export default src  }  declare module '\*.opus' {  const src: string  export default src  }  declare module '\*.mov' {  const src: string  export default src  }  declare module '\*.m4a' {  const src: string  export default src  }  declare module '\*.vtt' {  const src: string  export default src  }  // fonts  declare module '\*.woff' {  const src: string  export default src  }  declare module '\*.woff2' {  const src: string  export default src  }  declare module '\*.eot' {  const src: string  export default src  }  declare module '\*.ttf' {  const src: string  export default src  }  declare module '\*.otf' {  const src: string  export default src  }  // other  declare module '\*.webmanifest' {  const src: string  export default src  }  declare module '\*.pdf' {  const src: string  export default src  }  declare module '\*.txt' {  const src: string  export default src  }  // wasm?init  declare module '\*.wasm?init' {  const initWasm: (  options?: WebAssembly.Imports,  ) => Promise<WebAssembly.Instance>  export default initWasm  }  // web worker  declare module '\*?worker' {  const workerConstructor: {  new (options?: { name?: string }): Worker  }  export default workerConstructor  }  declare module '\*?worker&inline' {  const workerConstructor: {  new (options?: { name?: string }): Worker  }  export default workerConstructor  }  declare module '\*?worker&url' {  const src: string  export default src  }  declare module '\*?sharedworker' {  const sharedWorkerConstructor: {  new (options?: { name?: string }): SharedWorker  }  export default sharedWorkerConstructor  }  declare module '\*?sharedworker&inline' {  const sharedWorkerConstructor: {  new (options?: { name?: string }): SharedWorker  }  export default sharedWorkerConstructor  }  declare module '\*?sharedworker&url' {  const src: string  export default src  }  declare module '\*?raw' {  const src: string  export default src  }  declare module '\*?url' {  const src: string  export default src  }  declare module '\*?inline' {  const src: string  export default src  }  declare module '\*?no-inline' {  const src: string  export default src  }  declare module '\*?url&inline' {  const src: string  export default src  }  declare module '\*?url&no-inline' {  const src: string  export default src  }  declare interface VitePreloadErrorEvent extends Event {  payload: Error  }  declare interface WindowEventMap {  'vite:preloadError': VitePreloadErrorEvent  }  // index.d.mts  /// <reference types="node" />  import \* as nativeFs from "fs";  import picomatch from "picomatch";  //#region src/api/aborter.d.ts  /\*\*  \* AbortController is not supported on Node 14 so we use this until we can drop  \* support for Node 14.  \*/  declare class Aborter {  aborted: boolean;  abort(): void;  }  //#endregion  //#region src/api/queue.d.ts  type OnQueueEmptyCallback = (error: Error | null, output: WalkerState) => void;  /\*\*  \* This is a custom stateless queue to track concurrent async fs calls.  \* It increments a counter whenever a call is queued and decrements it  \* as soon as it completes. When the counter hits 0, it calls onQueueEmpty.  \*/  declare class Queue {  private onQueueEmpty?;  count: number;  constructor(onQueueEmpty?: OnQueueEmptyCallback | undefined);  enqueue(): number;  dequeue(error: Error | null, output: WalkerState): void;  }  //#endregion  //#region src/types.d.ts  type Counts = {  files: number;  directories: number;  /\*\*  \* @deprecated use `directories` instead. Will be removed in v7.0.  \*/  dirs: number;  };  type Group = {  directory: string;  files: string[];  /\*\*  \* @deprecated use `directory` instead. Will be removed in v7.0.  \*/  dir: string;  }; |