Тестовое задание на позицию FullStack-разработчик

1 Функциональные требования

Необходимо разработать систему учёта задач в соответствии с требованиями ниже.

Базовые требования

1. Разработать форму списка задач
   1.1 Вывести таблицу или список карточек задач. Отобразить поля: Название, Инициатор, Исполнитель, Статус, Срок.
   1.2 Добавить фильтры: статус (выпадающий список), Исполнитель
   1.3 Постраничный вывод элементов
   1.4 Дату просроченной задачи выделить красным цветом в интерфейсе. Просроченная задача определяется следующим образом:
   - Статус «Новая» или «В работе»
   - Срок < текущая дата (сравнение без учёта времени)
   1.5 Сортировка, желательно.

2. Разработать Формы создания, редактирования и просмотра задач.
   2.1 Добавить на формы поля в соответствии с Таблица 3. Описание полей (см столбцы Поле, Тип данных)
   2.2 Необходимо предусмотреть валидацию полей перед сохранением в соответствии с Таблица 3. Описание полей (см столбцы Правила заполнения).

Расширенные требования

3. Предусмотреть разграничение доступа на уровне отдельных Задач в соответствии с Таблица 1. Ролевая модель и Таблица 2. Таблица бизнес-процесс: статусы, действия, роли. Пользователь должен видеть в списке и иметь возможность открыть форму просмотра задачи только если ему предусмотрен доступ на просмотр.
   Основанные возможности, которые нужно показать:
   - В системе реализовано разграничение доступа не уровне объектов и ролевая модель, в рамках которой Инициатор и Исполнитель видят только свои Задачи

4. Бизнес-логика. Предусмотреть на форме просмотра Задачи кнопки-действия и разграничения по ролям в соответствии таблицей процесса. В т.ч. к функции редактирования. После выполнения действия на форме должны отображаться актуальные данные (обновлённая карточка)
   Основанные возможности, которые нужно показать:
   - В системе реализован бизнес-процесс: жизненный цикл/статусная модель Задачи
   - В системе бизнес-процессе реализована ролевая модель как в виде глобальных ролей, так и ролей в рамках конкретного экземпляра процесса (объекта)
   - Переводы статусов реализованы в виде команд, доступных в соответствии с ролевой моделью процесса

5. Предусмотреть разграничение доступ на уровне полей в соответствии с таблицей Таблица 3. Описание полей
   Основанные возможности, которые нужно показать:
   - В системе реализовано разграничение доступа на уровне отдельных полей.
   - Инициатором по умолчанию устанавливается текущий пользователь. Администратор может его поменять
   - Поля «Название», «Описание», «Срок», «Исполнитель» предназначены для Инициатора задачи, их может заполнять только Инициатор (или Администратор)
   - Поля «Отчет» предназначены для Исполнителя задачи, их может заполнять только Исполнитель (или Администратор)

Расширенные требования ++

6. Для действий по смене статуса (таких как В работу, Завершить и т.п.) предусмотреть всплывающее окно (ModalDialog), в котором отображается:
   - Название действия
   - Ввод комментария. Предусмотреть проверку на обязательность для негативных действий, таких как «Отменить» и «На доработку».
   - Кнопки подтверждения «Выполнить» (выполняется действие) или «Отменить» (просто закрываем окно)

7. Реализовать «Историю бизнес-процесса». При любом действии, повлекшем изменение состояния объекта (кроме Просмотра), необходимо делать запись в историю следующей информации:
   - Кто выполнил действие
   - Дата и время действия
   - ИД задачи
   - Название действия
   - Комментарий (если реализуется п 6.)
   Вывести историю в модальном окне (или на новой странице, или табе) в табличном виде при нажатии на кнопку История, расположенной на форме просмотра

8. Реализовать отправку уведомлений при смене статусов. Кому отправлять уведомления – нужно самостоятельно решить исходя из логики бизнес-процесса.

Таблица 1. Ролевая модель

Пользователь	Глобальная роль, любой аутентифицированный пользователь
Администратор	Глобальная роль, пользователь, прописанный в конфиге в списке администраторов
Инициатор	Роль по отношению к конкретной Задаче. Пользователь, который указан в поле Инициатор
Исполнитель	Роль по отношению к конкретной Задаче. Пользователь, который указан в поле Исполнитель

Таблица 2. Таблица бизнес-процесс: статусы, действия, роли

Статус	**Действие**	**Роль**	**Описание**
	Создание	Пользователь	Переводит в статус «Новая»
Новая	Редактировать	Админ Исполнитель	
	В работу	Админ Исполнитель	Переводит в статус «В работе» Установка поля «Дата начала» текущей датой, если оно пустое
	Отменить	Инициатор Админ	Переводит в статус «Отменена»
В работе	Редактировать	Админ Инициатор Исполнитель	
	Отменить	Инициатор Админ	Переводит в статус «Отменена»
	Завершить	Исполнитель Админ	Переводит в статус «Завершена» Установка поля «Дата завершения» текущей датой Действие недоступно (disabled), если не заполнено поле «Отчёт»
Завершена	Закрыть	Админ Инициатор	Переводит в статус «Закрыта»
	На доработку	Админ Инициатор	Переводит в статус «Новая»
Закрыта			
Отменена			
Все статусы	Просмотр	Админ Инициатор Исполнитель	

Таблица 3. Описание полей

Поле	**Тип данных**	**Правила заполнения**	**Доступность**
Название	Текст	Обязательно Минимум 5 символов	Просмотр - только при создании или статусе Новая - только роли Инициатор или Администратор
Описание	Многострочный текст	Обязательно	Просмотр - только при создании или статусе Новая - только роли Инициатор или Администратор
Инициатор	Справочник	Обязательно По умолчанию заполняется текущим пользователем	Просмотр Редактирование: - только роли Администратор
Исполнитель	Справочник	Обязательно	Просмотр Редактирование:

			- только при создании или статусе Новая - только роли Инициатор или Администратор
Срок	Дата	Обязательно	Просмотр - только при создании или в статусе «Новая» - только роли Инициатор или Администратор
Отчёт	Многострочный текст	Обязательно Минимум 5 символов	Просмотр - только в статусе «В работе» - только роли Исполнитель или Администратор
Статус	Справочник		Просмотр
Дата начала	Дата		Просмотр (если не пустое)
Дата завершения	Дата		Просмотр (если не пустое)

2 Технические требования:

Желательно использовать следующий технологический стек:

- Frontend: любой современный Framework (Angular, React, Vue или др.)
- UI-kit/Controls: желательно что-то использовать, например Bootstrap (или аналог), библиотеки контролов (Material, Prime, Telerik, DevExpress)
- Backend на ASP.NET Core
- DataAccess/ORM: любой современный Entity Framework, Dapper, Hibernate
- Data: любая БД (желательно MS SQL) и проект БД внутри солюшена (чтобы показать умение делать схемы БД)

3 Требования к предоставлению результата

Для предоставления результата необходимо:

- Разместить код в доступном из интернет репозитории (github или др.) и прислать ссылку
- Развернуть систему для просмотра в интернете (или возможность запустить у себя на Demo-звонке)
- Предоставить беклог: список задач, примерная декомпозиция задач с указанием оценок (план/факт), как выполненных, так и будущих задач (которые можно было бы выполнить в следующей итерации). Желательно первый вариант беклога составить вначале выполнения задания. Можно в Excel или другом простом формате

4 Требования к документации

Создать документацию в виде .md файла в репозитории или Word файла и описать в нём:

- Реализованные бизнес-требования (можно скопировать из соответствующего раздела) в соответствии с выполненным объёмом
- Инструкцию по установке: клонируем репозиторий, делаем восстановление пакетов, пеняем конфиги, разворачиваем БД, запускаем и т.п.

- Краткую инструкцию пользователя (несколько сценариев/операций со скриншотами)
- Если предполагается вход под разными пользователями указать как их добавить/переключать
- Предложения по рефакторингу кода и улучшению.

5 Критерии оценки

При оценке задания будет обращено внимание на следующие моменты:

- Объем выполненных задач, их работоспособность, соответствие требованиям, наличие видимых багов
- Качество решение: внешний вид, качество UI, удобство, user friendly, адаптивность к различным разрешениям, отсутствие явной «кривизны».
- Используемый стек: современность, обоснованность выбора библиотек и т.п.
- Качество кода: структура проекта, читаемость кода, стиль кода, аккуратность и т.п.
- Архитектура: понимание и использование шаблонов проектирования, способы реализация отдельных классов/уровней (Repository, Service, DataAccess, Api, Workflow, Dependency Injection и т.п.) и универсальность, расширяемость, дублирование кода
- Использование базовых подходов в приложении: логирование, конфигурационные файлы, обротка исключений
- Качество документации
- Демонстрация решения на Demo
- Умение работать с репозиторием (частота коммитов, комментарии к коммитам)
- Умение работать с беклогом (декомпозиция задач, оценки)

6 Рекомендации

При выполнении тестового задания лучше следовать рекомендациям:

- Договориться о максимальном времени на задание, стараться не превышать его. Нет цели сделать всё, но за определённое время нужно сделать что-то рабочее, и с нормальным качеством кода.
- Требования располагаются в порядке приоритета, можно делать из по порядку. Базовые требования в приоритете.
- Оставить немного времени на документирование, беклог и др. задачи.
- Нужно показать текущие знания и использовать уже знакомые подходы, инструменты и библиотеки, т.к. есть риск не успеть разобраться и не уложиться во времени, в итоге ничего не сделать

Хороший продуманный код дольше писать и отлаживать, а время на задание ограничено, как и в реальном проекте. Если есть желание показать большее умение и понимание, то можно сделать следующее:

- Сделать адекватный минимум, но выделить немного больше времени на отдельные 1-2 функции/области, затем при личном общении пояснить, что было сделано, почему так сделано, и как это можно перенести на другие части приложения
- Составить список слабых сторон текущей реализации, список улучшений/рефакторинга и суметь о них рассказать.
- Добавить рекомендации по рефакторингу в виде комментариев TODO непосредственно в тех участках, которые нужно доработать/улучшить.
- Подумать над теми требованиями, которые не получилось сделать и быть готовым обсудить способы реализации.