Тестовое задание для программиста C#.

Данное тестовое задание служит для выявления навыков программиста работы с требуемыми технологиями (работа с потоками, логирование, работа с БД, Dependency Injection, unit-тестирование). Поэтому не следует искать в нём практический смысл. Возможно те же задачи можно реализовать другим способом. Но интересна способность кандидата применять именно указанные технологии.

Задание

Требуется создать решение (солюшн), состоящее как минимум из двух проектов.

Проект 1

WPF приложение.

Главная форма - содержит таблицу и кнопки, либо другие элементы управления, которые буду описаны ниже. Дополнительная форма содержит таблицу. При запуске приложения запускаются два дополнительных потока (Task или Thread).

В первом потоке, с периодом 2 секунды (с момента запуска приложения) выбирается очередной элемент массива автомобилей (строка), скажем (Мондео, Крета, Приус, УАЗик, Вольво, Фокус, Октавия, Запорожец и т.д.) и генерируется элемент {Метка времени, название автомобиля}.

Во втором, с периодом 3 секунды (с момента запуска приложения) выбирается очередной элемент массива водителей (строка), скажем (Петр, Василий, Николай, Марина, Феодосий, Карина и т.д.) и генерируется элемент {Метка времени, имя водителя}.

Метки времени должны быть согласованы. Т.е. таймер должен быть общий на приложение, и таким образом, каждые 6 секунд метки времени должны совпадать у автомобилей и водителей.

Генерируемые элементы должны записываться в БД. Автомобили в свою таблицу, водители в свою.

На главной форме должна отображаться сводная таблица водителей и автомобилей (как совпавших, так и не совпавших). Все, с начала запуска приложения. Без чтения данных из БД. Столбцы: метка времени, автомобиль, водитель.

С помощью кнопок или элементов управления на главной форме необходимо предоставить возможность прервать или запустить снова любой из указанных потоков.

Также должна быть кнопка открытия дополнительной формы, которая содержит таблицу. Содержимое таблицы считывается из БД и представляет из себя сводную таблицу по метке времени всех элементов обеих таблиц, как совпавших по времени, так и не совпавших. Содержимое таблицы должно обновляться либо раз в секунду, либо по добавлению новых элементов в таблицы. Чтобы постоянно видеть новые элементы в таблице, её содержимое должно быть отсортировано по метке времени в обратном порядке, либо (при сортировке в прямом порядке) курсор всегда должен находиться на самой нижней строке. Дополнительная форма должна открываться не модально, т.е. не блокировать доступ к основной форме.

События совпадения элементов, запуска/останова приложения, запуска/останова потоков, открытия/закрытия дополнительных форм должны записываться в лог файл, там же, где

находится приложение. Каждые 5 минут должен создаваться новый лог файл со временем создания (5-минуткой) и датой в названии.

Доступ к функциональным классам классам/интерфейсам (работа БД, логирование, работа с потоками и т.п.), необходимо сделать посредством внедрения зависимостей (Dependency Injection in .NET).

Проект 2

Проект Unit – тестов. Чем больше кода покрыто тестами, тем лучше. Желательно сделать тест, который проверял бы количество сгенерированных водителей и машин за какой-то промежуток времени. Например, за 4 секунды, за 6 секунд.

Конечный результат

Результат выполнения задания: солюшн с исходными текстами программы и тестов, если БД не создаётся EntityFramework при первом запуске, то нужен скрипт создания БД. Желательно использовать SQLite или SQL Express.