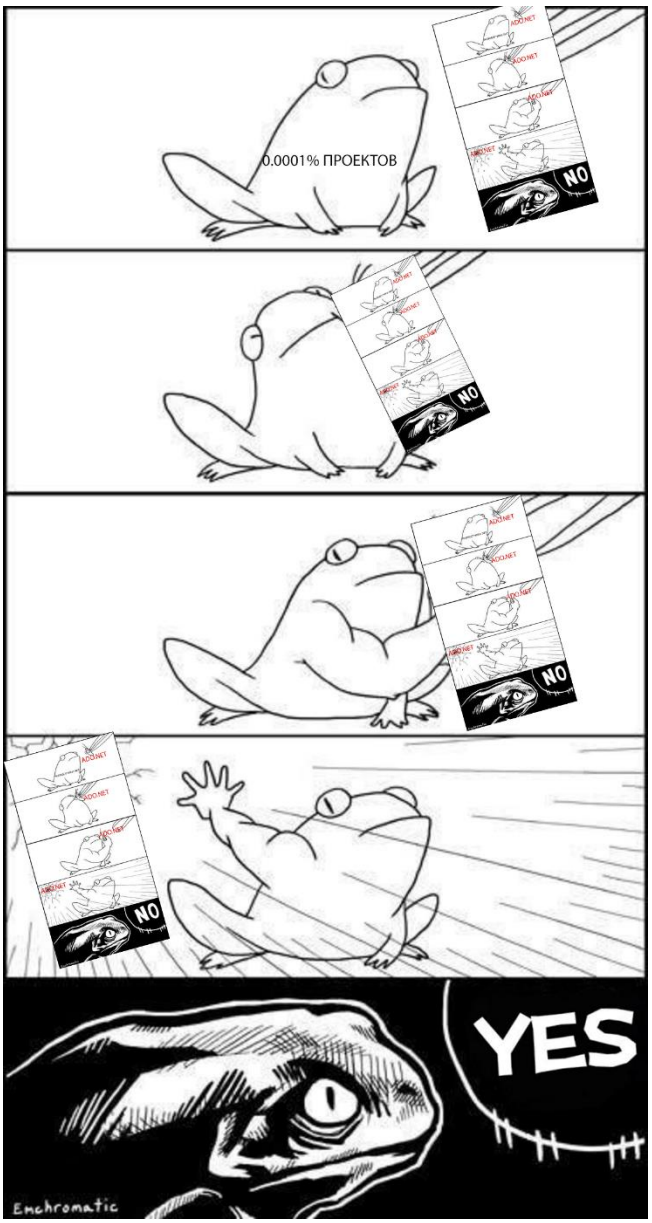

Практическая работа № 2

Так, ладно. Архитектура оказалась несколько непонятной штукой. Но дальше вы, вероятно, начнёте разбираться, и будет легче!

В архитектуру, созданную в первой работе, внести следующие изменения:

- ✅ Добавить **DataAccessLayer** – слой обеспечивающий работу с данными. Создайте там интерфейс IRepository, обеспечивающий основные CRUD операции (Create, Read, ~~Update~~, Delete). И реализуйте два сценария работы с СУБД: через EF и Dapper.
- ✅ **Model** – слой, который хранит доменные объекты, изменить таким образом, чтобы экземпляры можно было сохранить в базу данных.
- ✅ **BusinessLogic** – слой в котором находится реализация основной логики приложения (бизнес логика). Изменить слой так, чтобы он работал с базой данных.
- ✅ **View** – слой, отвечающий за вывод списка сотрудников и предоставляющий возможность добавить/удалить сотрудника (редактирование – Update - пока можно не реализовывать). По-прежнему для WinForms есть возможность отрисовать гистограмму.

Пояснения:

1. Реализуем интерфейс IDomainObject. В интерфейсе описываем свойство ID. Доменные классы должны реализовать этот интерфейс
2. Создаем интерфейс IRepository, обеспечивающий основные CRUD операции (Create, Read, ~~Update~~, Delete)
3. Подключаем через управление пакетами NuGet:
    a. EntityFrameWork
    b. Dapper
4. Создаем dll библиотеку DataAccessLayer
5. В этой библиотеке описываем 2 сценария работы с БД
    a. EF сценарий. Для его реализации создаем классы:
        i. DBContext
        ii. EntityRepository
    b. Dapper сценарий. Для его реализации создаем 1 класс:
        i. DapperRepository
6. Ваши 2 репозитория должны имплементировать интерфейс IRepository
7. В классе Business Logic создаем объект:
   IRepository repository = new….
8. Уровень view, по факту, не меняется.