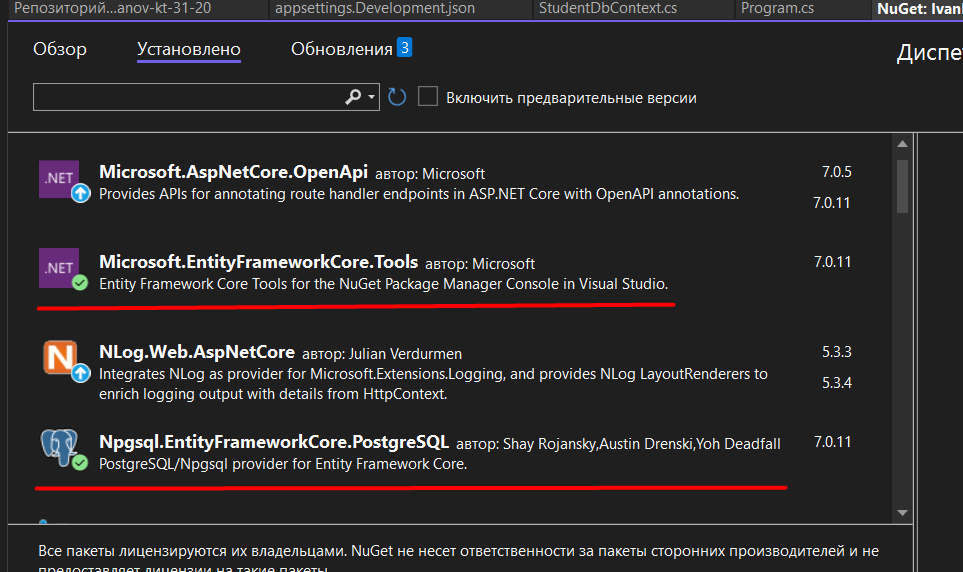
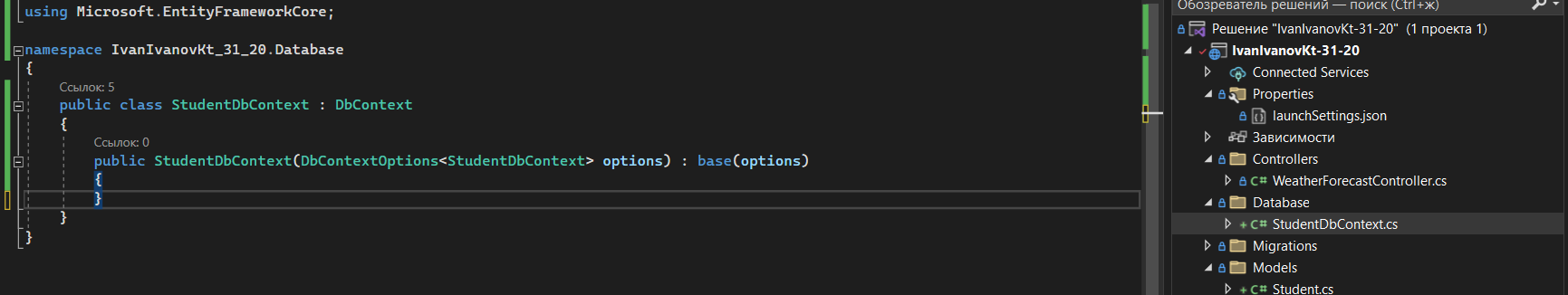
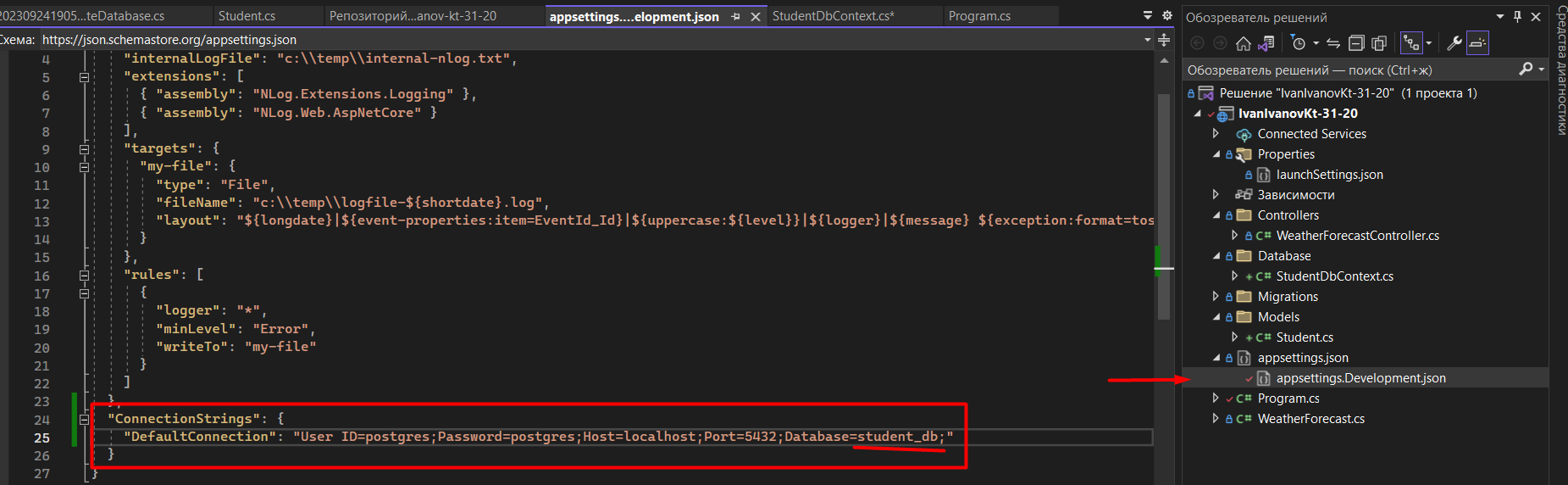
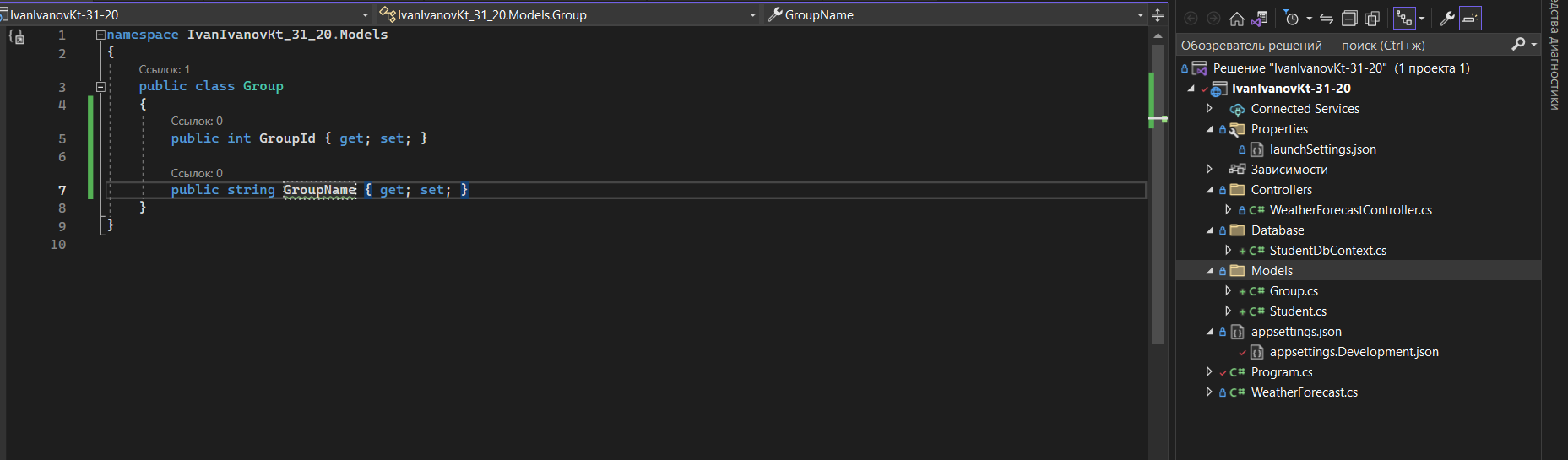
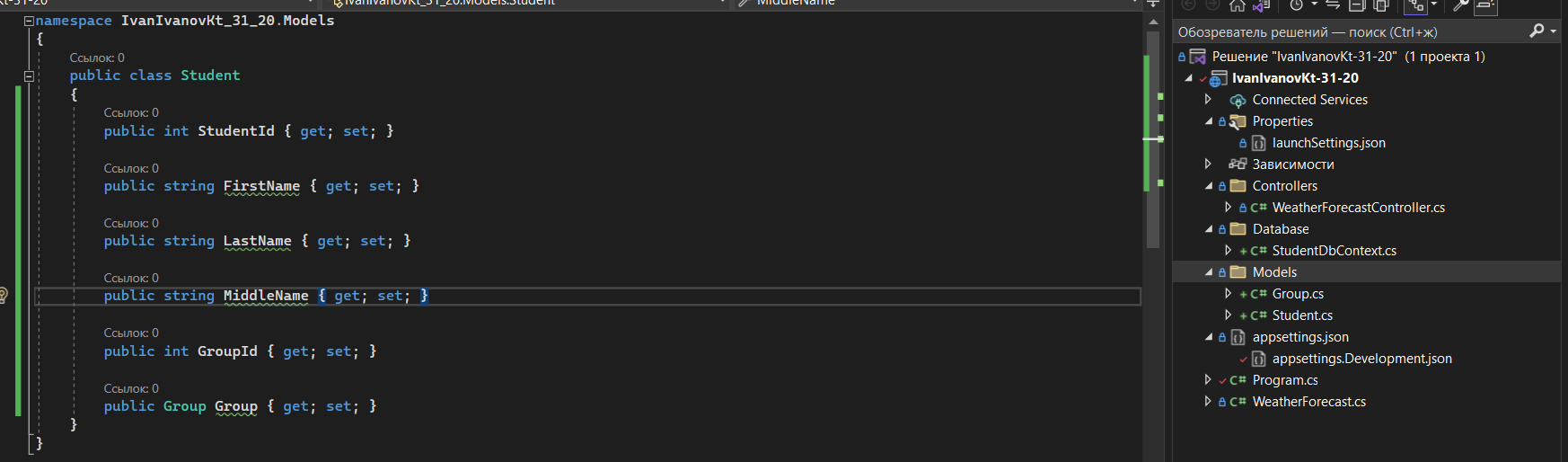
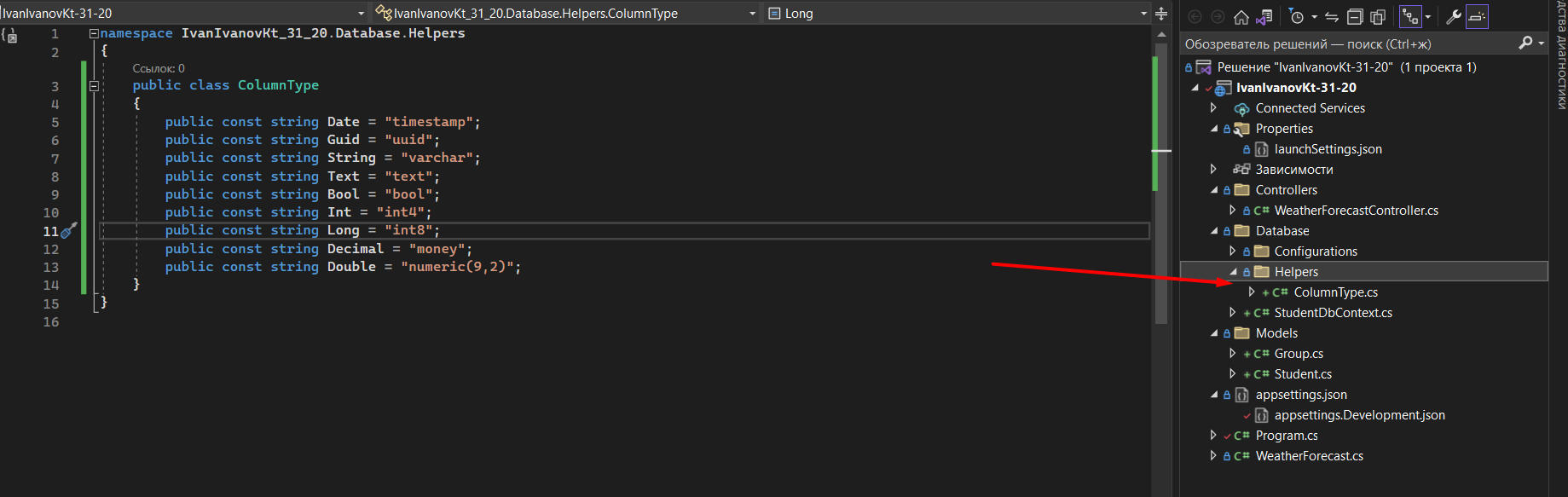
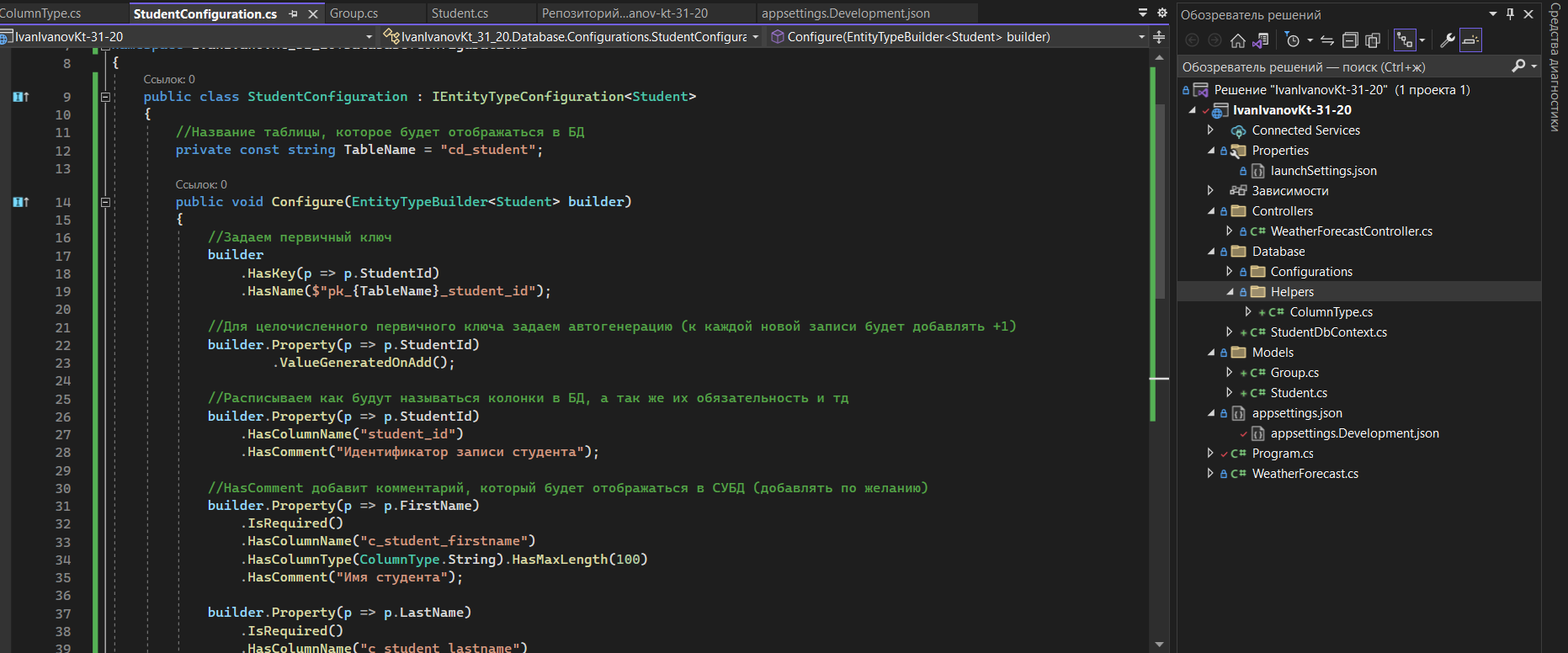
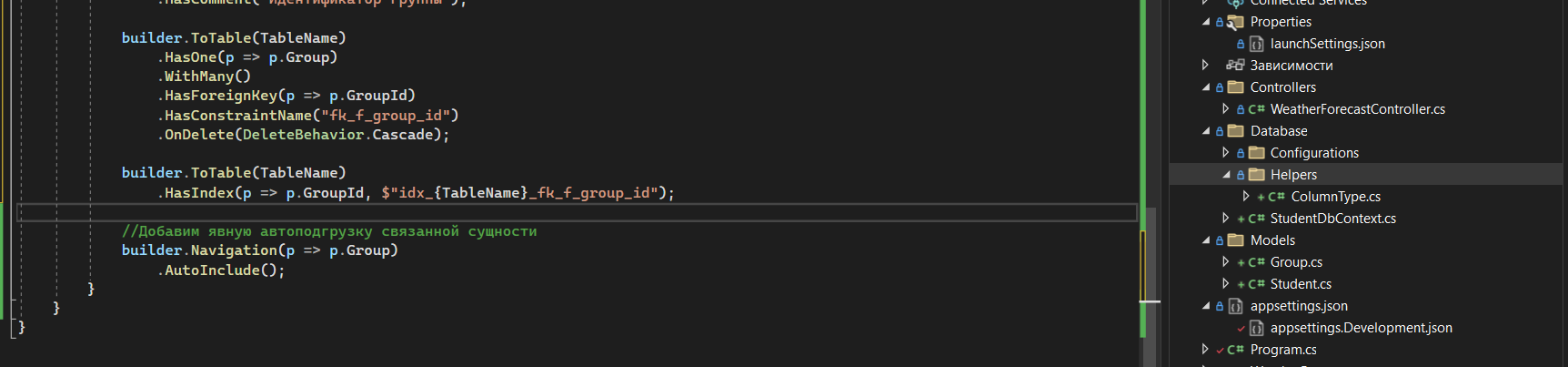
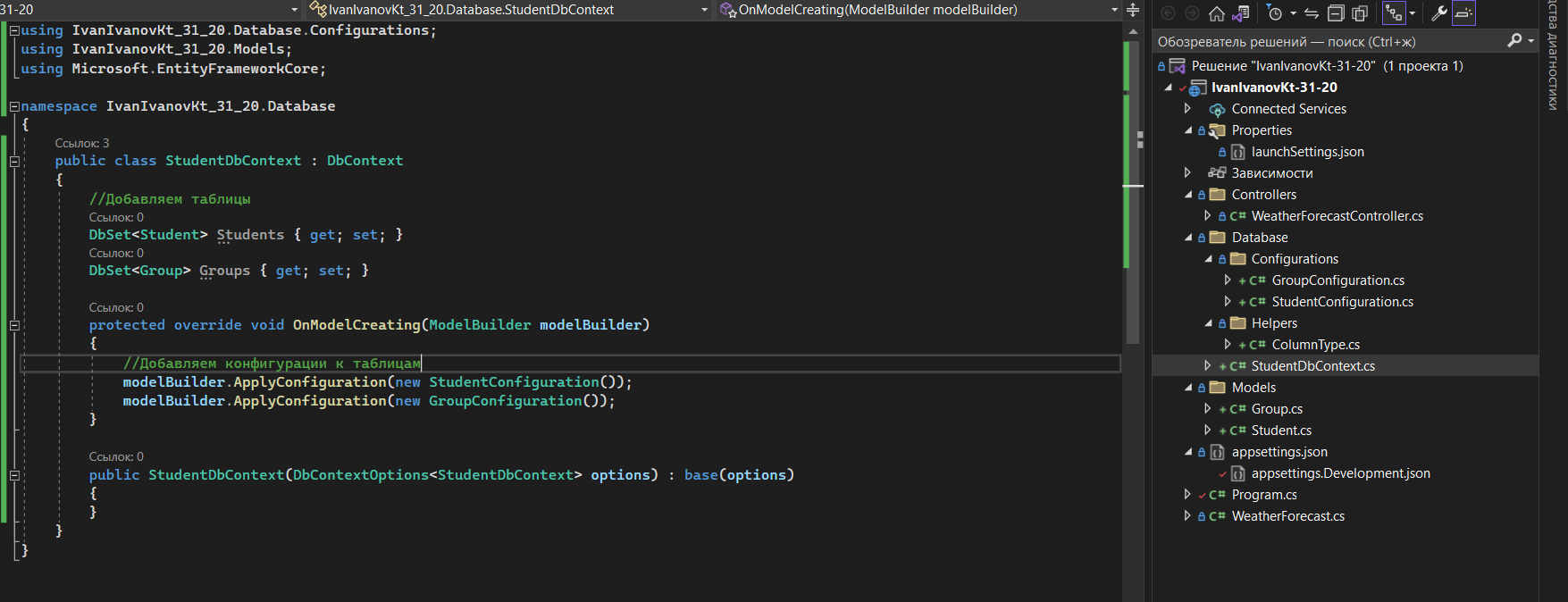
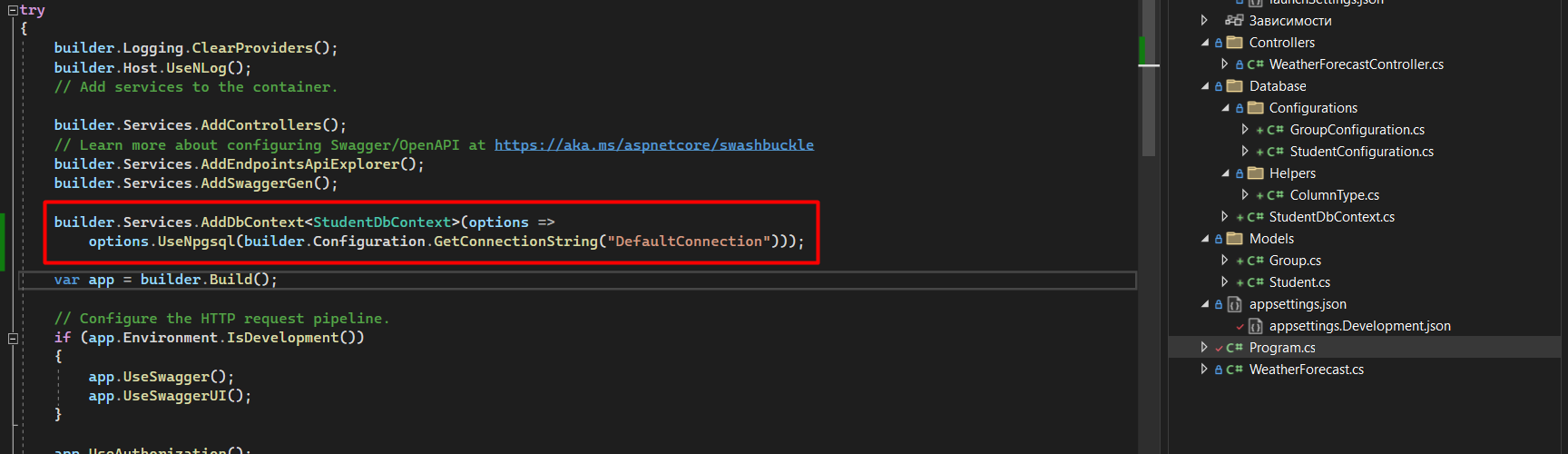
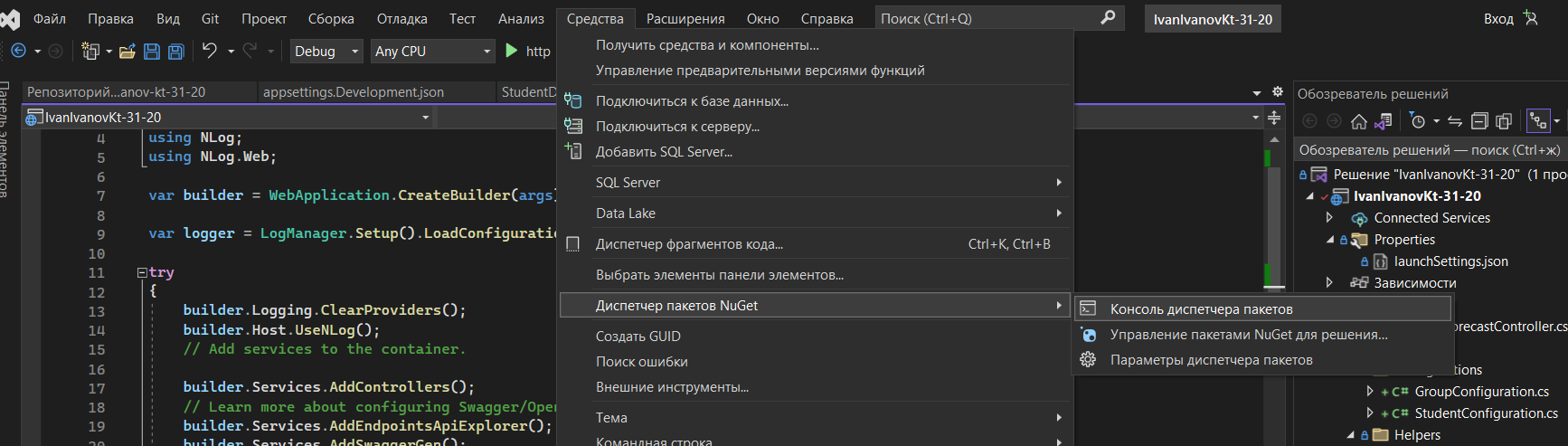
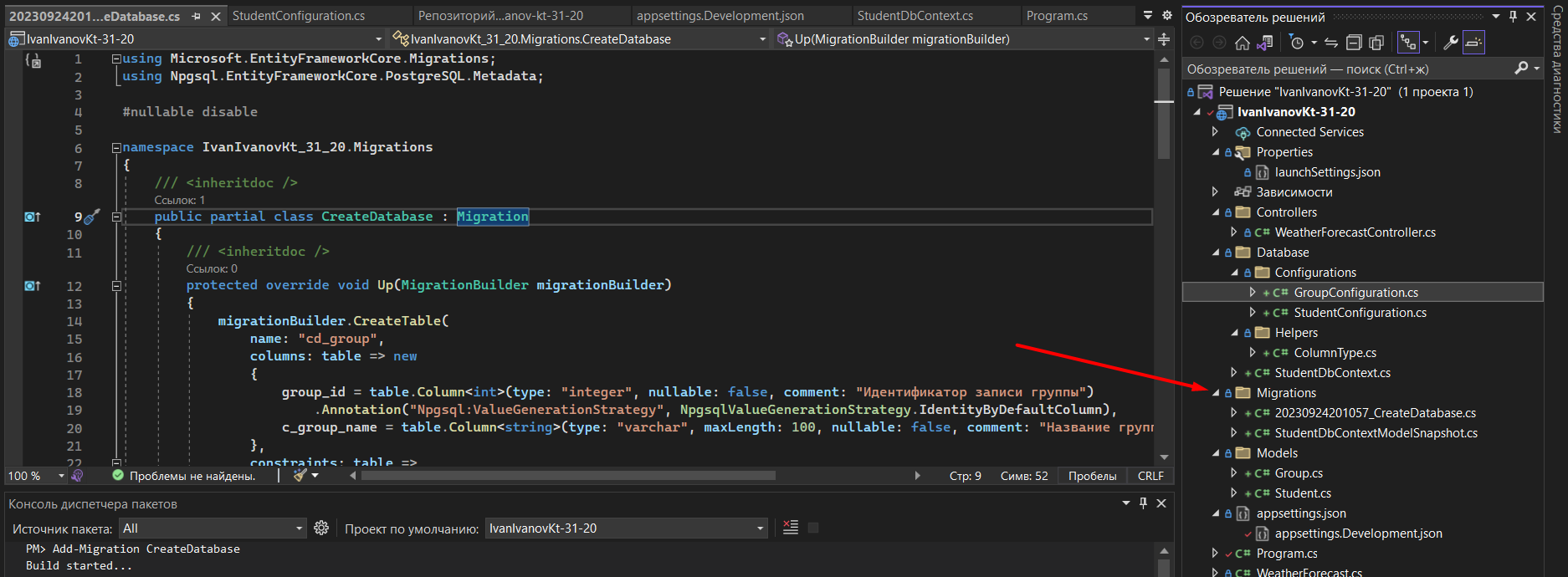
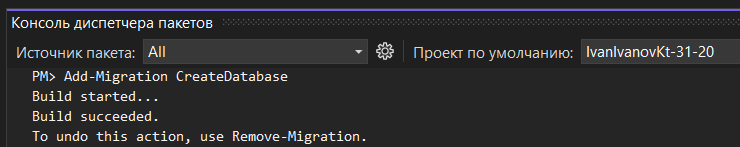
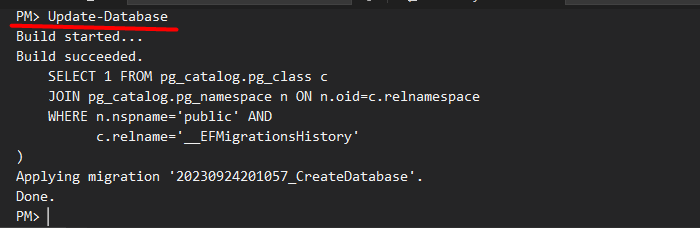
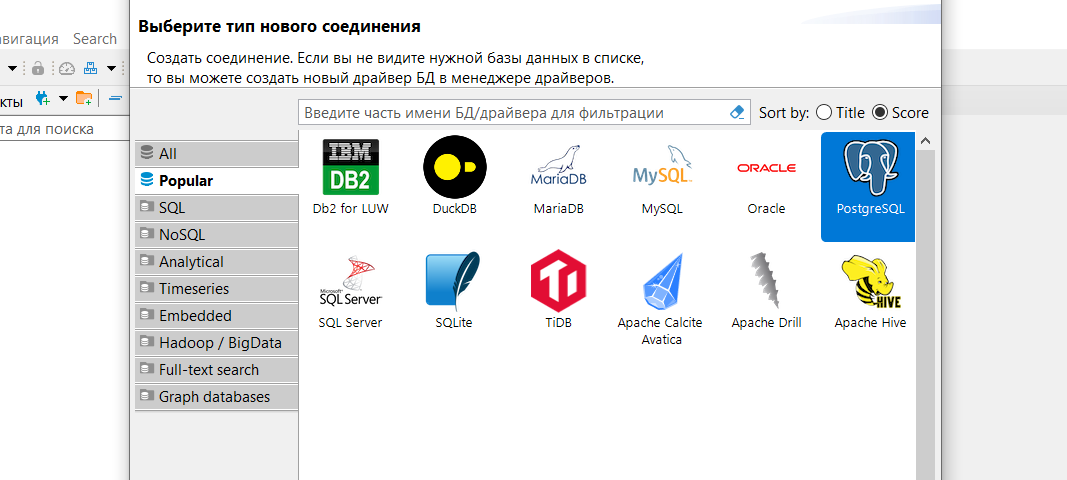
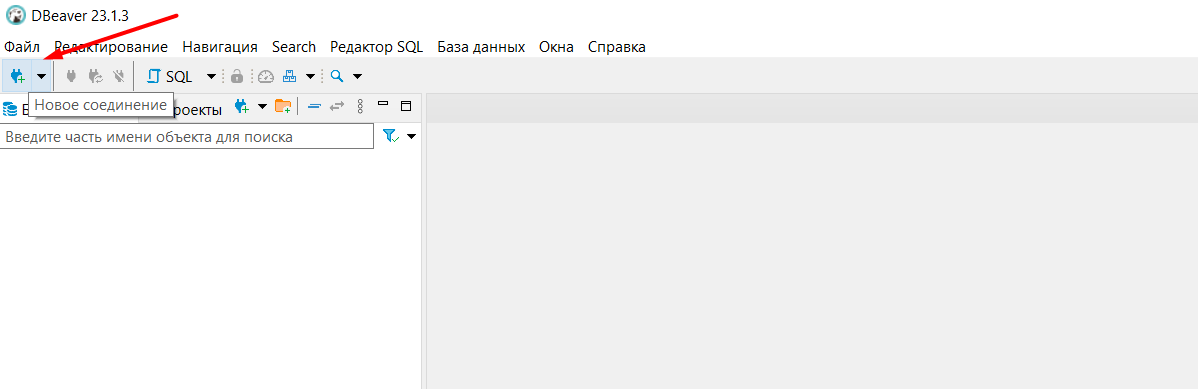
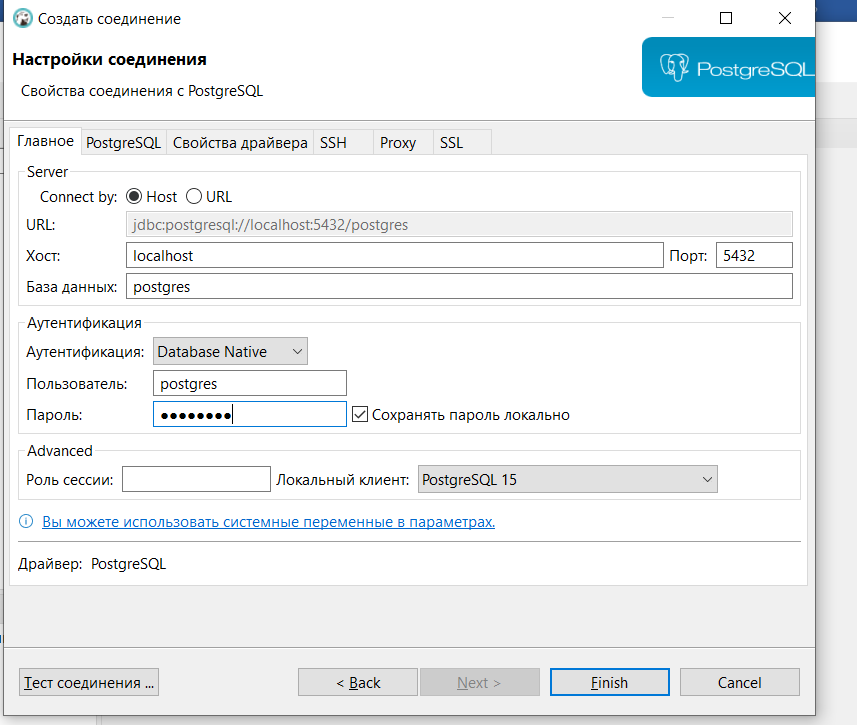
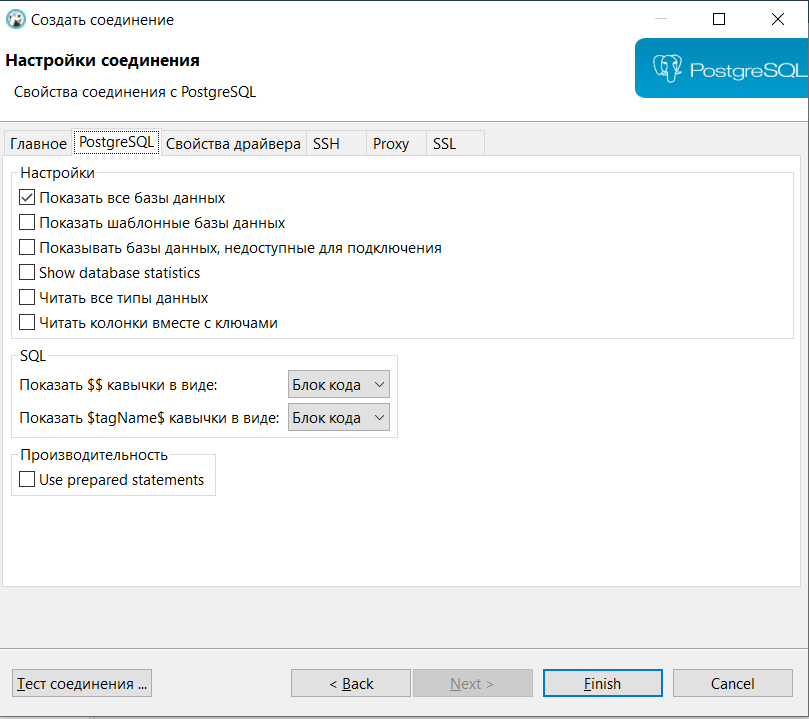
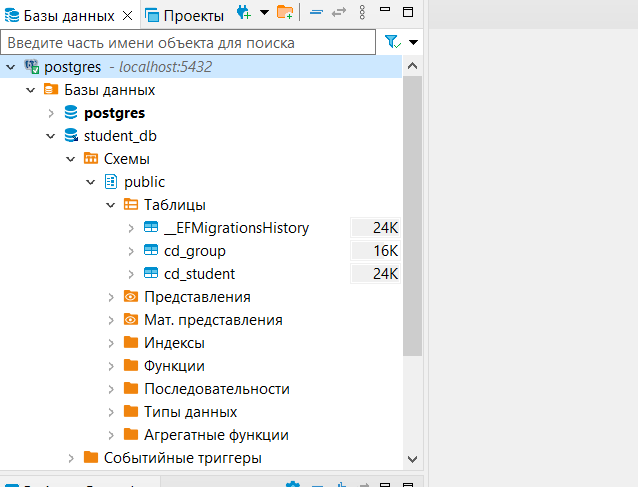
Лабораторная работа 3

Добавление контекста базы данных, моделей(таблиц), миграций

В примере будет использована БД PostgreSQL и СУБД DBeaver. Можно использовать БД SQL Server и СУБД SQL Server Management Studio

1. Установить ядро PostgreSQL <https://www.postgresql.org/download/windows/> если не установлено
2. Установить СУБД DBeaver <https://dbeaver.io/download/>
3. Добавить Nuget пакеты, если используете SQL Server, то вместо пакета Npgsql… использовать Microsoft.EntityFrameworkCore.SqlServer
4. Добавить папку и класс DbContext. Название класса выбирайте по своему усмотрению, отталкиваясь от варианта
5. Прописать строку подключения в appsettings.Development.json, название базы данных пишите по своему усмотрению (название контекста базы данных и самой базы данных, это разные вещи, учитывайте и не путайтесь) 
6. Создать модели для БД. Они будут охарактеризовывать таблицы БД. Для примеры я буду отталкивать от варианты сервиса «Получение списка студентов с фильтрацией по группе». Очевидно, что нужна таблица Студенты и таблица Группы, которые будут связаны между собой внешним ключом. Так как отношение Группы -> Студенты будет как Один ко многим, следует добавить в модель Student свойство GroupId и навигационное свойство Group. GroupId в дальнейшем будет использоваться как столбец для внешнего ключа, а навигационное свойство, позволит получать доступ сразу к данным связанной группы (связанные данные можно подгружать тремя способами Eager Loding, Explicit Loading, Lazy Loading, подробнее читайте в интернете) 
7. Для более удобного конфигурирования создать класс-хелпер
8. Созданные модели, просто создадут таблицы в бд, без всякой связанности, для конфигурирования индексов, ключей и тд. В современном подходе следует использовать Fluent API подход, когда каждая отдельная таблица конфигурируется отдельно и явно, это помогает в больших проектах следить за состоянием БД, из минусов, замедляет разработку
9. До этого были расписаны правила для колонок, теперь будут описаны связи. В данном случае код описывает, что у таблицы Студентов есть внешний ключ на Группы и группа для Студентов одна, в свою очередь у каждой Группы может быть много студентов, поэтому получаем HasOne WithMany. Далее указывается свойство в которое будет использоваться в качестве внешнего ключа, а так же название зависимости. OnDelete описывает, какой сценарий будет происходить со Студентами при удалении Группы. Я подразумеваю, что студент не может сущестовать без группы, поэтому ставлю ограничение на каскадное удаление. Другие свойства SetNull и тд можете посмотреть сами и использовать для других видов зависимостей. В случае определения трудностей задавайте вопросы преподавателю.
10. После создания всех конфигураций необходимо добавить изменения в контекст
11. Теперь необходимо прописать контекст БД в Program.cs и подключить его через строку подключения, указанной в пунтке 6
12. Создадим миграцию. Миграции используются для версионирования БД, в случае плохо написанной миграции, БД всегда можно будет откатить к предыдущей. Необходимо вызвать Консоль диспетчера пакетов. 
13. В консоли диспетчера пакетов пишем команду Add-Migration CreateDatabase (CreateDatabase – это название миграции, для дальнейших миграций используйте названия описывающе смысл миграции) В случае успеха, будет отображен такой текст, а в проект добавится папка Migrations. Прочитайте текст миграции, команды описаные здесь, пойдут в обновление БД. Метод UP описывает изменения , которые создает миграция. В методе Down описываются изменения, которые произведутся в случае отката миграции на более ранюю версию.
14. После того, как убедились, что всё успешно, необходимо накатить миграцию на БД. Для этого в консоли диспетчера пакетов необходимо прописать команду Update-Database. 
15. Проверим, что всё действительно создалось в БД. Для этого открываем Dbeaver и подсоединямся к созданной БДВо вкладке пароль пишите postgresВо вкладке PostgreSQL выбираем галочку Показать все базы данных и жмем Finish
16. Раскрыть «выпадайки» и проверить что все таблицы и колонки, которые вы описывали – на месте. Таблица \_\_EFMigrations… содержит в себе миграции, можно не обращать внимания на текущий момент
17. В случае успеха. Закоммитить изменения