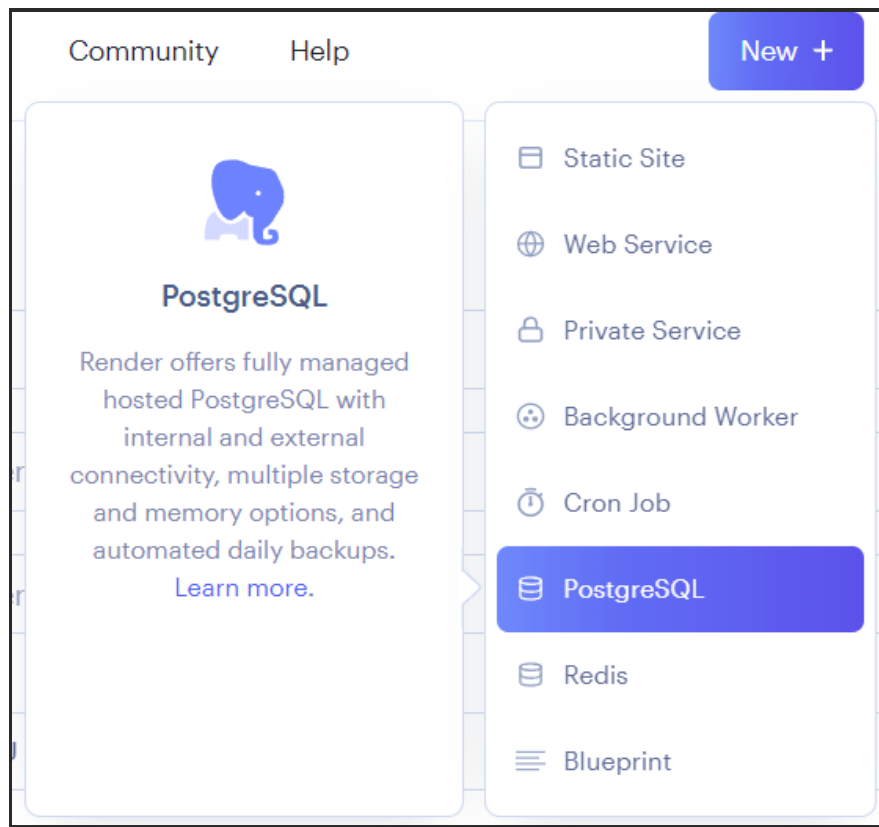


### 3. Создание доменной модели


В результате выполнения лабораторной работы должна быть спроектирована модель данных, описывающая выбранную прикладную область..

Для выполнения данной лабораторной работы вам необходимо выбрать поставщика данных уровня баз данных. Рекомендуется воспользоваться любым Database as a Service решением, либо разрешается использовать любую выделенную реляционную СУБД. Далее будет рассмотрен вариант с подключением базы данных от хостинга Render.com.

Для того чтобы использовать базу данных Postgres в качестве СУБД необходимо выбрать соответствующий пункт в Dashboard панели хостинга Render



**Обратите внимание** на то, что сервис Render.com предоставляет бесплатный доступ к СУБД в ознакомительном режиме

 Free databases will expire in 90 days and will be deleted if not upgraded. Learn more about [free instance type limits](#).

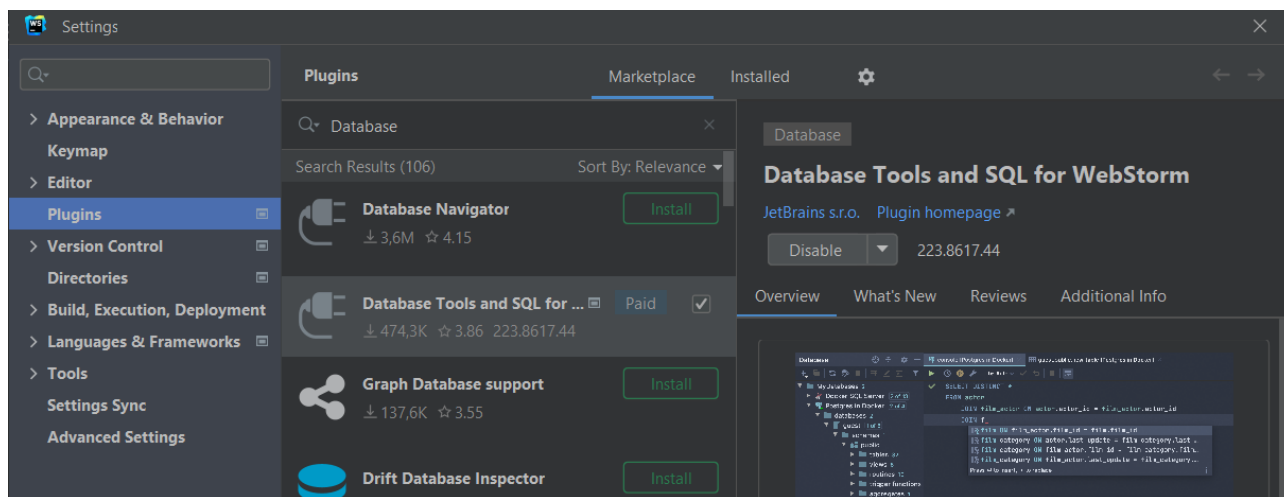
С момента создания базы данных у вас будет **3 месяца** для того чтобы сделать все последующие лабораторные работы. По истечению трёх месяцев база данных будет автоматически удалена.

Параметры создания базы данных предложенные Wizard'ом можно не менять, пользователь и название самой базы будет сгенерировано автоматически. В случае если вы хотите что-то кастомизировать, то это можно сделать согласно [документации](#) представленной хостингом.

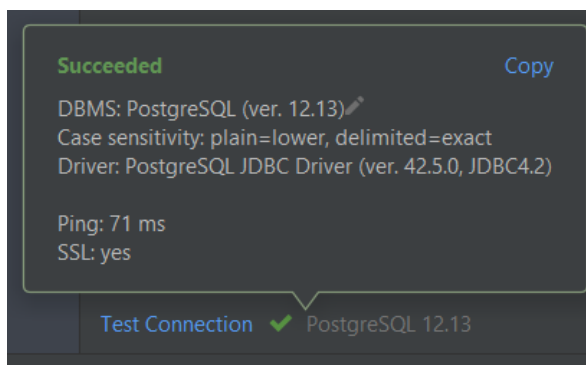
После того как вы создадите экземпляр СУБД, вам будет предоставлено два URL'а. Один, который вы должны использовать непосредственно в задеплоенном на `render.com` приложение и второй, который вы можете использовать для подключения со своей локальной машины.

Hostname	dpg-c1mn43n9re0mg9deh6fg
Port	5432
Database	test_database_nis1
Username	test_database_nis1_user
Password	<input type="password"/>
Internal Database URL	<input type="password"/>

Проверьте что вы можете подключиться к созданной базе данных. Рекомендую использовать инструмент встроенный в WebStorm инструмент дублирующий функционал DataGrip'а. По-умолчанию он не включен в дистрибутив, поэтому [установите пакет из маркетплейса самостоятельно](#).

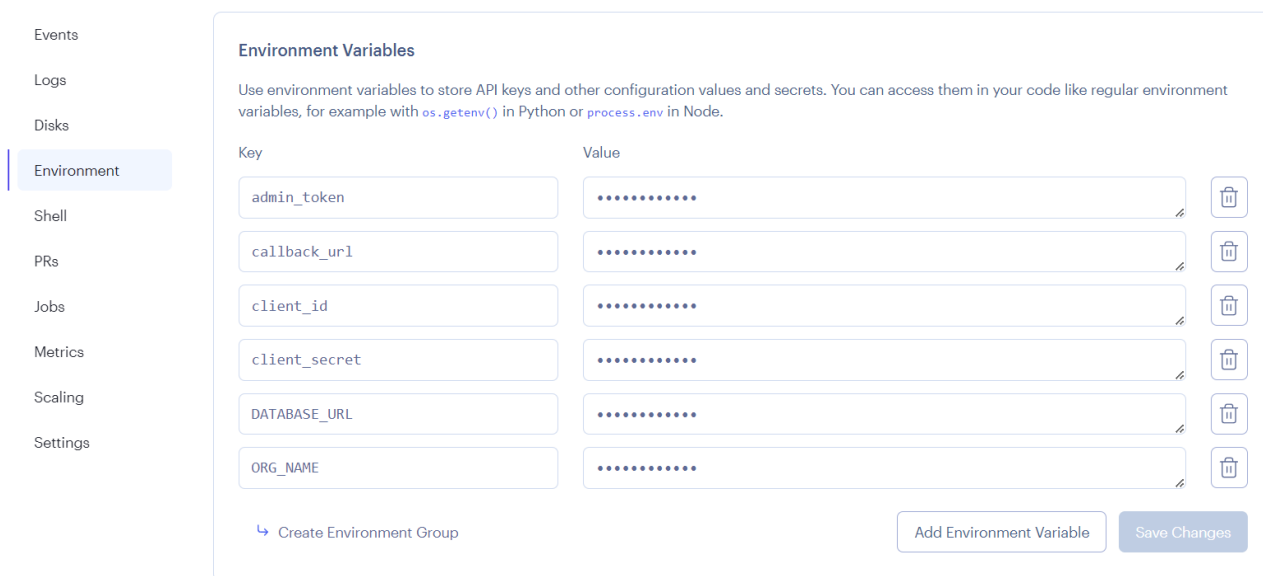


Подключение к Postgres не должно сильно отличаться от MSSQL, с которым вы уже имели опыт работы на курс ранее, просто в качестве клиента вы будете использовать IDE а не отдельный инструмент.



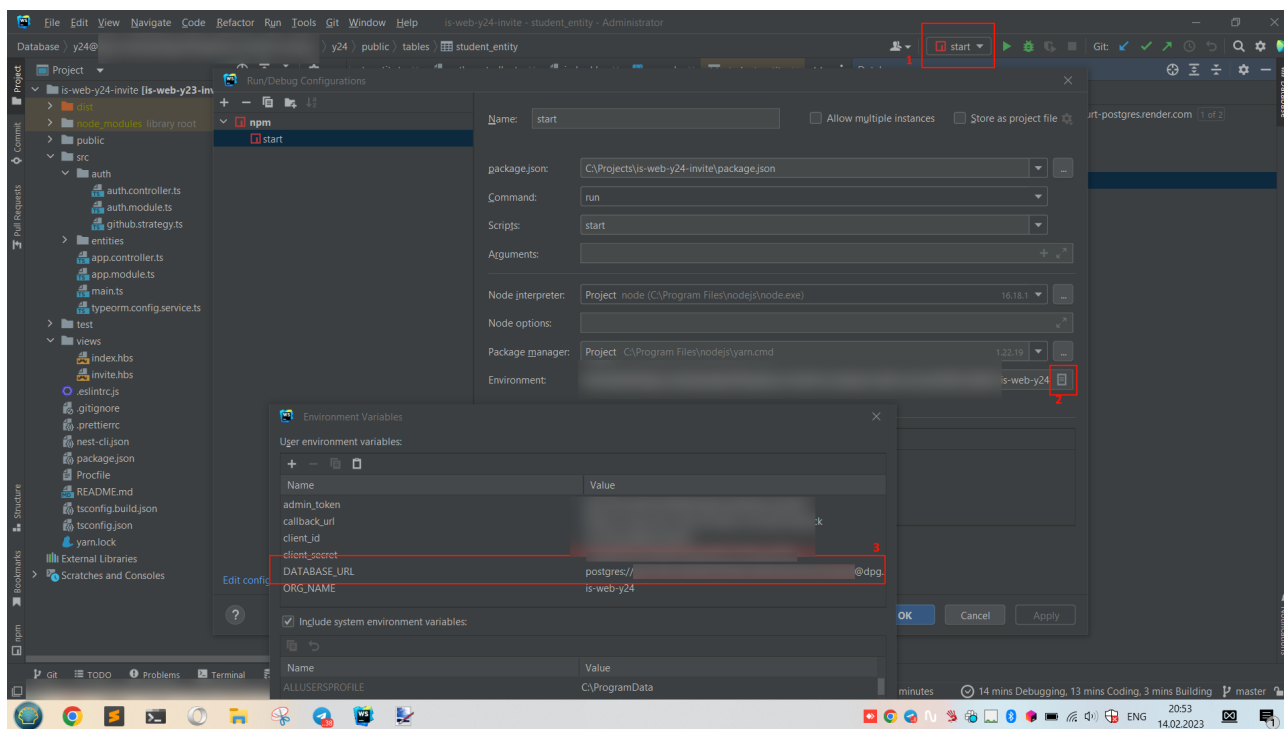
Если у вас получилось получить сообщение о том что подключение успешно установлено, то можно переходить к следующему шагу - подключению СУБД к вашему проекту.

Первым делом, необходимо добавить Connection String как переменную среды окружения для вашего приложения. Для этого зайдите в раздел **Environment** и создайте ключ **DATABASE\_URL** как на изображении ниже



Значение для этого ключа скопируйте из поля “Internal Database URL”.

Также добавьте соответствующий External Database URL для локальной конфигурации в ваш проект как на скриншоте ниже



т.к. используется Postgres, то необходимо добавить пакет 'pg' как зависимость (pg это ODBC для node.js):

```
npm install pg --save
```

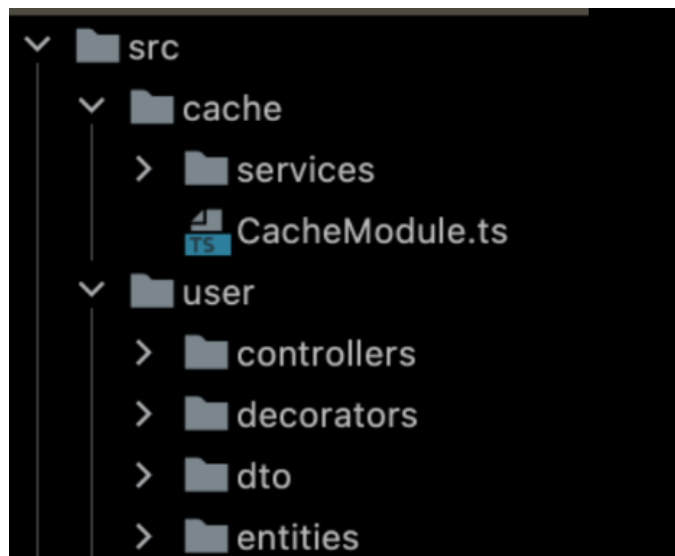
Далее необходимо выбрать одну из поддерживаемых ORM для работы со слоем данных. Рекомендуется либо [TypeORM](#) либо [Prisma](#).

В случае, если вы используете TypeORM, вам понадобится распарсить connectionString к базе данных и предоставить её ORM'ке в том виде, как это описано в документации, в виде отдельных полей - Host, username, password и т.д. Для решения этой задачи рекомендую воспользоваться npm пакетом `pg-connection-string`. Ознакомьтесь с разделом [Async Configuration](#) и реализуйте сервис инфраструктурного слоя DDD для создания объекта необходимого ORM для подключения.

В случае, если вы используете Prisma, то connection string можно использовать as-is. В качестве инфраструктурного слоя необходимо создать файл schema.prisma и добавить в него указания поставщика и названия ключа переменной окружения - `env(DATABASE_URL)`

После подключения ORM, необходимо описать все модели необходимые для работы вашего проекта. Рекомендуется описывать сущности согласно DDD с использованием общего языка выбранной предметной области.

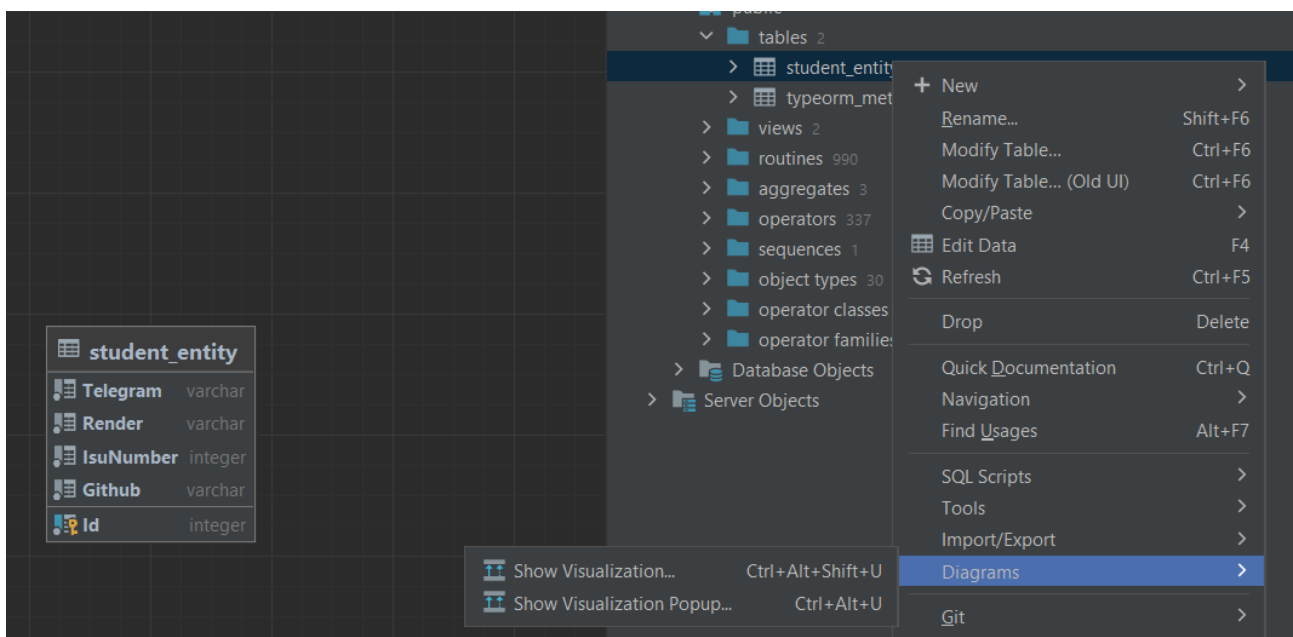
Например, на примере UserModule из лекционного материала, если вы создаёте модуль для работы с пользователем, то выделяйте сущности по смыслу в отдельные директории (src/%moduleName%/entities/%entityName%.ts)

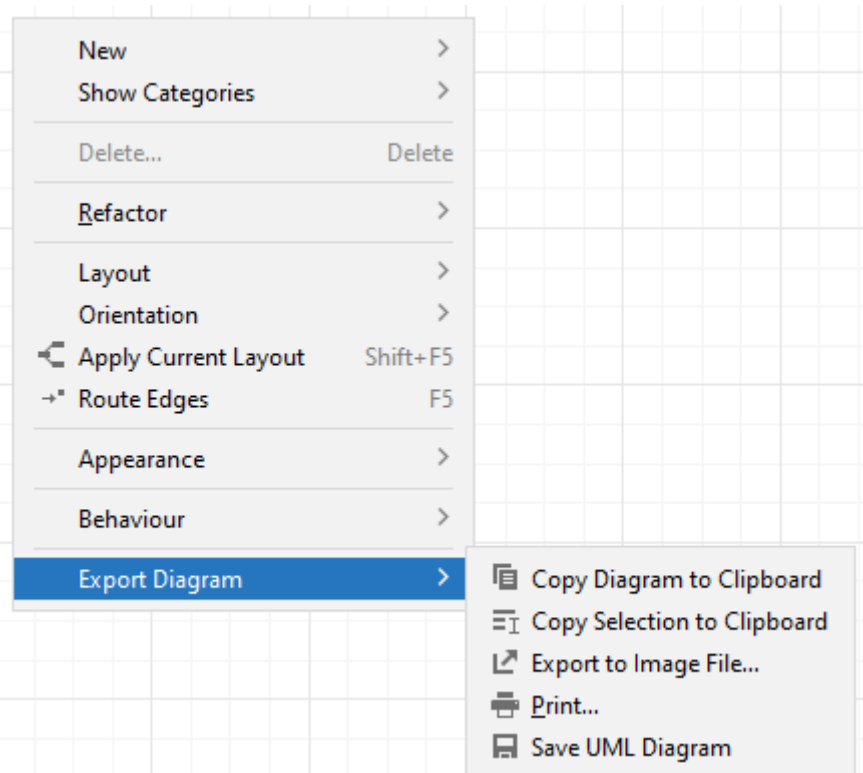


Для того чтобы ваши сущности были обработаны TypeOrm'ом необходимо добавить декоратор [@Entity](#) согласно документации. Если в конфигурации ORM вы добавили поле `synchronize` со значением `true` то согласно правилу описанному в поле `"entities": ["src/**/*.entity{.ts,.js}"]` все ваши сущности с этим декоратором будут автоматически мигрированы в базу данных.

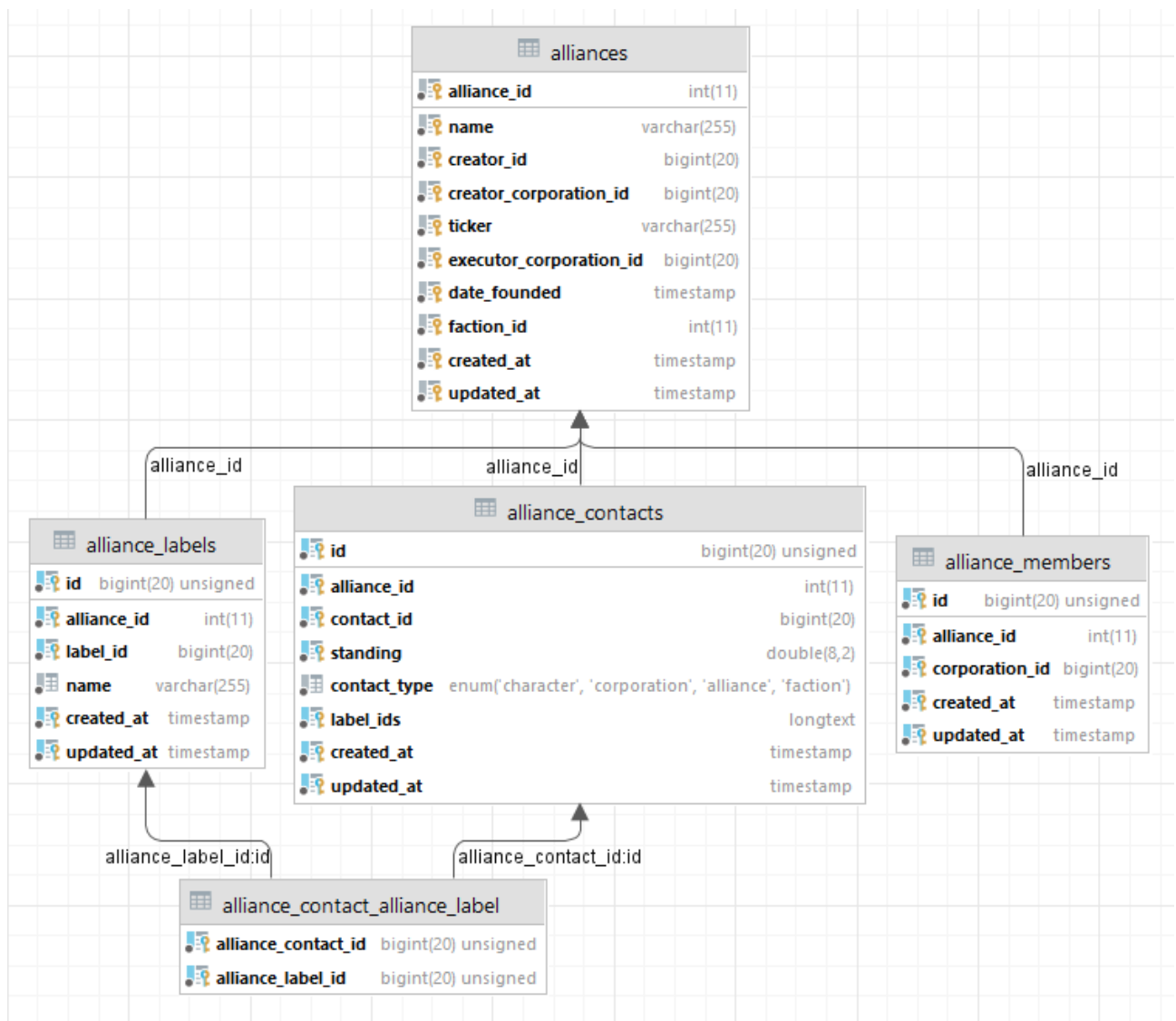
Для Prisma ORM схема объявляется отдельно [согласно документации](#).

После создания всех моделей необходимо сгенерировать визуальное представление, см. пример ниже:





Сгенерируйте визуальное представление схемы вашей данных в виде ERD (диаграммы). Пример диаграммы в DataGrip представлен ниже:



Полученную схему в виде ERD диаграммы загрузите в корень вашего репозитория и опишите значение сущностей (текстовое описание на русском языке) в файле [readme.md](#)