**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

# **Дисциплина:** Бэк-энд разработка

Отчет

Домашняя работа #2

Выполнил:

Игнатьев Алексей

К3340

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

**Задание:**

Реализовать все модели данных, спроектированные в рамках ДЗ1

Реализовать набор из CRUD-методов для работы с моделями данных средствами Express + TypeScript

Реализовать API-эндпоинт для получения пользователя по id/email

**Ход работы**

1. **Проектирование моделей в Prisma**

В рамках данного пункта были описаны все модели базы данных в Prisma, включая отношения между таблицами, а так же корректный naming таблиц в postgresql

1. **CRUD методы**

Для таблиц, в которых требуются CRUD методы – реализованы данные методы. В некоторых случаях использован расширенный CRUD, с внутренней агрегацией и дополнительными полями

1. **API-Endpoint**

Реализован метод получения пользователя по его Access-токену. В Access токене зашит ID пользователя, по нему вытаскивается информация о пользователе

**Вывод**

* Описаны модели таблиц в Prisma Orm
* Реализованы CRUD методы для необходимых таблиц
* Сделана авторизация и получение информации о пользователе по его ACCESS токену

// This is your Prisma schema file,  
// learn more about it in the docs: https://pris.ly/d/prisma-schema  
  
// Looking for ways to speed up your queries, or scale easily with your serverless or edge functionscategoriesry Prisma Accelerate: https://pris.ly/cli/accelerate-init  
  
generator client {  
 provider = "prisma-client-js"  
}  
  
datasource db {  
 provider = "postgresql"  
 url = env("DATABASE\_URL")  
}  
  
enum UserAliasType {  
 EMAIL  
}  
  
enum PaymentMethod {  
 CARD  
 CRYPTO  
}  
  
enum PaymentStatus {  
 CREATED  
 EXPIRED  
 COMPLETED  
}  
  
enum AuctionState {  
 PAYMENT  
 DELIVERY  
 OBTAINING  
 REVIEW  
}  
  
enum AuctionStatus {  
 NOT\_STARTED  
 ACTIVE  
 COMPLETED  
}  
  
enum AuctionType {  
 OPENED  
 CLOSED  
}  
  
enum NotificationType {  
 AUCTION\_REMINDER // Напоминание за 30 минут  
 AUCTION\_STARTED // Аукцион начался  
 AUCTION\_WON // Победа в аукционе  
 AUCTION\_LOST // Поражение в аукционе  
 AUCTION\_STATUS\_CHANGE // Изменение статуса победителя  
 BALANCE\_UPDATE // Изменение баланса  
 REFERRAL\_BALANCE\_UPDATE // Изменение баланса реферала  
}  
  
model News {  
 news\_id Int @id @default(autoincrement())  
 image\_url String?  
 title String  
 text String  
 created\_at DateTime @default(now())  
  
 @@map("news")  
}  
  
model Faq {  
 faq\_id Int @id @default(autoincrement())  
 question String  
 answer String  
  
 @@map("faqs")  
}  
  
model Settings {  
 settings\_id Int @id @default(autoincrement())  
 cost\_per\_spin Int @default(0)  
  
 @@map("settings")  
}  
  
enum RatoTaskType {  
 INVITE\_FRIEND  
 MAKE\_BID  
 SHARE\_PLATFORM  
}  
  
enum RewardType {  
 RATO  
 FREE\_SPIN  
}  
  
model RatoTask {  
 rato\_task\_id Int @id @default(autoincrement())  
 data String?  
 type RatoTaskType  
 reward\_type RewardType  
 reward\_amount Float  
 created\_at DateTime @default(now())  
 updated\_at DateTime @updatedAt  
 user\_tasks UserTask[]  
  
 @@map("rato\_tasks")  
}  
  
model UserTask {  
 user\_task\_id Int @id @default(autoincrement())  
 user\_id Int  
 task\_id Int  
 completed\_at DateTime?  
 user User @relation(fields: [user\_id], references: [user\_id])  
 task RatoTask @relation(fields: [task\_id], references: [rato\_task\_id])  
  
 @@unique([user\_id, task\_id])  
 @@map("user\_tasks")  
}  
  
enum SpinItemType {  
 RATO  
 OTHER  
}  
  
model SpinItem {  
 spin\_item\_id Int @id @default(autoincrement())  
 title String?  
 description String?  
 image\_url String?  
 amount Int?  
 probability Float  
 is\_active Boolean @default(false)  
 type SpinItemType  
 fortune\_wins FortuneWins[]  
  
 @@map("spin\_items")  
}  
  
model FortuneWins {  
 fortune\_win\_id Int @id @default(autoincrement())  
 spin\_item\_id Int  
 user\_id Int  
 created\_at DateTime @default(now())  
 is\_received Boolean @default(false)  
 user User @relation(fields: [user\_id], references: [user\_id])  
 spin\_item SpinItem @relation(fields: [spin\_item\_id], references: [spin\_item\_id])  
  
 @@map("fortune\_wins")  
}  
  
model Transaction {  
 transaction\_id Int @id @default(autoincrement())  
 payment\_id String?  
 user\_id Int  
 method PaymentMethod  
 status PaymentStatus  
 currency String? @db.VarChar(10)  
 amount Float  
 exchange\_rate Float?  
 rato\_amount Float?  
 created\_at DateTime @default(now())  
 expired\_at DateTime?  
 user User @relation(fields: [user\_id], references: [user\_id])  
  
 @@map("transactions")  
}  
  
enum BotStrategyType {  
 LOW  
 MIDDLE  
 HIGH  
 BOSS  
}  
  
model BotStrategy {  
 bot\_strategy\_id Int @id @default(autoincrement())  
 type BotStrategyType  
  
 // Размеры ставок  
 min\_bid\_amount Int? // Минимальная ставка  
 max\_bid\_amount Int? // Максимальная ставка  
  
 // Интервалы между ставками  
 min\_bid\_interval Int? // Минимальный интервал между ставками в секундах  
  
 // Количество участников  
 min\_auction\_participants Int? // Минимальное количество участников аукциона  
 max\_auction\_participants Int? // Максимальное количество участников аукциона  
 min\_active\_participants Int? // Минимальное количество участников со ставками  
 max\_active\_participants Int? // Максимальное количество участников со ставками  
  
 // Размеры RATO при перебитии  
 min\_rato\_increase Int? // Минимальный размер RATO при перебитии  
 max\_rato\_increase Int? // Максимальный размер RATO при перебитии  
  
 // Временные ограничения  
 time\_from\_start Int? // Время от старта (начало)  
 time\_to\_start Int? // Время от старта (конец)  
 time\_from\_end Int? // Время до окончания (начало)  
 time\_to\_end Int? // Время до окончания (конец)  
  
 bots Bot[]  
  
 @@map("bot\_strategies")  
}  
  
model Bot {  
 bot\_id Int @id @default(autoincrement())  
 strategy\_id Int  
 user\_id Int @unique  
 is\_active Boolean @default(true)  
 auction\_id Int?  
 strategy BotStrategy @relation(fields: [strategy\_id], references: [bot\_strategy\_id])  
 auction Auction? @relation(fields: [auction\_id], references: [auction\_id])  
 user User @relation(fields: [user\_id], references: [user\_id])  
  
 @@map("bots")  
}  
  
model User {  
 user\_id Int @id @default(autoincrement())  
 created\_at DateTime @default(now())  
 updated\_at DateTime @updatedAt  
 rato\_balance Float @default(0.00)  
 user\_aliases UserAlias[]  
 first\_name String?  
 last\_name String?  
 free\_spin\_amount Int @default(0)  
 instagram String?  
 phone\_number String?  
 username String?  
 mail\_promo Boolean @default(true)  
 mail\_new\_auctions Boolean @default(true)  
 mail\_rate\_result Boolean @default(true)  
 mail\_confirmed Boolean @default(false)  
 language String?  
 referral\_code String? @unique  
 email String?  
 avatar\_url String?  
 auction\_participants AuctionParticipant[]  
 transactions Transaction[]  
 won\_auctions Auction[] @relation("AuctionWinner")  
 review Review[]  
 referrals Referral[] @relation("UserReferrals")  
 invited\_by Referral? @relation("UserInvitedBy")  
 notification Notification[]  
 favorites Favorite[]  
 address Address?  
 user\_tasks UserTask[]  
 fortune\_wins FortuneWins[]  
 bot Bot?  
  
 @@map("users")  
}  
  
model Referral {  
 id Int @id @default(autoincrement())  
 user\_id Int @unique  
 referred\_by Int  
 created\_at DateTime  
 user User @relation("UserInvitedBy", fields: [user\_id], references: [user\_id])  
 referrer User @relation("UserReferrals", fields: [referred\_by], references: [user\_id])  
  
 @@index([referred\_by])  
 @@map("referrals")  
}  
  
model Address {  
 address\_id Int @id @default(autoincrement())  
 user\_id Int @unique  
 user User @relation(fields: [user\_id], references: [user\_id], onDelete: Cascade)  
 country String?  
 city String?  
 street String?  
 house\_number String?  
  
 @@map("addresses")  
}  
  
model UserAlias {  
 user\_alias\_id Int @id @default(autoincrement())  
 user\_id Int  
 alias\_type UserAliasType @default(EMAIL)  
 value String @db.VarChar(255)  
 proof String @db.VarChar(255)  
  
 user User @relation(fields: [user\_id], references: [user\_id])  
  
 @@map("user\_aliases")  
}  
  
model AuctionBrand {  
 auction\_brand\_id Int @id @default(autoincrement())  
 title String  
 auction Auction[]  
  
 @@map("auction\_brands")  
}  
  
model AuctionCategory {  
 auction\_category\_id Int @id @default(autoincrement())  
 title String  
 auction Auction[]  
  
 @@map("auction\_categories")  
}  
  
model AuctionImage {  
 auction\_image\_id Int @id @default(autoincrement())  
 url String  
 auction\_id Int  
 auction Auction @relation(fields: [auction\_id], references: [auction\_id], onDelete: Cascade)  
  
 @@map("auction\_images")  
}  
  
model Auction {  
 auction\_id Int @id @default(autoincrement())  
 created\_at DateTime @default(now())  
 updated\_at DateTime @updatedAt  
 start\_time DateTime  
 start\_price Float  
 end\_price Float  
 reservation\_price Float?  
 rate\_step Float?  
 color String @db.VarChar(7)  
 title String  
 auction\_number String?  
 rate\_time Int?  
 announce\_time DateTime  
 description String?  
 discount String?  
 free\_spin\_amount Int @default(0)  
 max\_participants Int?  
 end\_time DateTime?  
 state AuctionState  
 is\_draft Boolean @default(false)  
 brand\_id Int  
 category\_id Int?  
 winner\_id Int?  
 status AuctionStatus  
 type AuctionType  
 auction\_participants AuctionParticipant[]  
 category AuctionCategory? @relation(fields: [category\_id], references: [auction\_category\_id])  
 winner User? @relation(fields: [winner\_id], references: [user\_id], name: "AuctionWinner")  
 brand AuctionBrand @relation(fields: [brand\_id], references: [auction\_brand\_id])  
 review Review?  
 images AuctionImage[]  
 favorites Favorite[]  
 bots Bot[]  
  
 @@map("auctions")  
}  
  
model AuctionParticipant {  
 auction\_participant\_id Int @id @default(autoincrement())  
 auction\_id Int  
 user\_id Int  
 rate Int?  
 expired\_rate DateTime?  
 created\_at DateTime @default(now())  
 updated\_at DateTime @updatedAt  
 auction Auction @relation(fields: [auction\_id], references: [auction\_id], onDelete: Cascade)  
 user User @relation(fields: [user\_id], references: [user\_id])  
  
 @@map("auction\_participants")  
}  
  
model Review {  
 review\_id Int @id @default(autoincrement())  
 auction\_id Int @unique  
 user\_id Int  
 media\_url String  
 media\_type MediaType  
 comment String  
 created\_at DateTime @default(now())  
  
 user User @relation(fields: [user\_id], references: [user\_id])  
 auction Auction @relation(fields: [auction\_id], references: [auction\_id])  
  
 @@map("reviews")  
}  
  
enum MediaType {  
 PHOTO  
 VIDEO  
}  
  
model Notification {  
 notification\_id Int @id @default(autoincrement())  
 user\_id Int  
 type NotificationType  
 message String  
 created\_at DateTime @default(now())  
 value Float? // Используется для пополнения баланса (опционально)  
  
 user User @relation(fields: [user\_id], references: [user\_id])  
  
 @@map("notifications")  
}  
  
model Favorite {  
 user\_id Int  
 auction\_id Int  
 user User @relation(fields: [user\_id], references: [user\_id])  
 auction Auction @relation(fields: [auction\_id], references: [auction\_id])  
  
 @@id([user\_id, auction\_id])  
 @@map("favorites")  
}  
  
enum LocalizationType {  
 BRAND\_TITLE  
 CATEGORY\_TITLE  
 NEWS\_TITLE  
 NEWS\_TEXT  
 FAQ\_QUESTION  
 FAQ\_ANSWER  
}  
  
model Localization {  
 locale\_id Int @id @default(autoincrement())  
 language String // Код языка, например "en", "ru"  
 text String  
 type String  
 reference\_id Int  
  
 // Уникальность: одно имя и описание для одной сущности и одного языка  
 @@unique([locale\_id, type])  
 @@map("localizations")  
}