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

2. **CRUD методы**

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

3. **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
}
```

```prisma
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: [auc-
tion_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], refer-
ences: [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: [auc-
tion_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], ref-
erences: [auction_category_id])
  winner               User?               @relation(fields: [winner_id], refer-
```

```
ences: [user_id], name: "AuctionWinner")
  brand                AuctionBrand         @relation(fields: [brand_id], refer-
ences: [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: [us-
er_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")
}
```