**Документация к функции createOrder**

Функция createOrder предназначена для добавления нового заказа на покупку билетов в базу данных. Включает проверку уникальности события, уникальности штрихкода и бронирование заказа через внешний API (с использованием моков). Если бронирование и подтверждение успешны, заказ сохраняется в базе данных.

**Параметры функции**

* **$event\_id** (int) — уникальный идентификатор события.
* **$event\_date** (string) — дата и время события в формате Y-m-d H:i:s.
* **$ticket\_adult\_price** (int) — цена билета для взрослого на момент покупки.
* **$ticket\_adult\_quantity** (int) — количество взрослых билетов.
* **$ticket\_kid\_price** (int) — цена детского билета.
* **$ticket\_kid\_quantity** (int) — количество детских билетов.

**Процесс выполнения**

1. **Подключение к базе данных**:
   * Устанавливается соединение с MySQL.
   * В случае ошибки подключение завершается.
2. **Проверка на уникальность event\_id**:
   * Функция проверяет, существует ли уже заказ с таким event\_id.
   * Если заказ существует, возвращается сообщение об ошибке без добавления нового заказа.
3. **Генерация уникального штрих-кода (barcode)**:
   * Генерируется случайное число от 10000000 до 99999999.
   * Проверяется уникальность barcode в базе данных, чтобы избежать дублирования.
   * В случае дублирования генерация повторяется, пока не будет найден уникальный штрих-код.
4. **Попытка бронирования через мок API**:
   * Отправляется запрос на бронирование с использованием функции mockBookingAPI.
   * Если ответ API содержит ошибку barcode already exists, генерируется новый barcode.
   * Если бронирование завершилось ошибкой (кроме barcode already exists), функция возвращает сообщение об ошибке.
   * Если сгенерировался уникальный штрихкод, сохраняем его в переменную и идём дальше.
5. **Подтверждение бронирования через мок API**:
   * Если бронирование успешно, отправляется запрос на подтверждение с помощью mockApprovalAPI.
   * В случае успешного подтверждения рассчитывается общая стоимость заказа, после чего он сохраняется в базе данных.
6. **Сохранение заказа в базу данных**:
   * Данные заказа сохраняются в таблице nevatrip\_order с использованием подготовленного запроса.
   * Вставляется информация о event\_id, event\_date, ticket\_adult\_price, ticket\_adult\_quantity, ticket\_kid\_price, ticket\_kid\_quantity, barcode, equal\_price, created (текущая дата и время).
   * Если все шаги выполнены, функция возвращает строку с сообщением об успешном создании заказа и его штрихкодом.

**Возвращаемое значение**

* **string** — сообщение о результатах выполнения:
  + Успех: "Заказ успешно создан с штрихкодом: [barcode]"
  + Ошибка: "Ошибка при бронировании: ..." или "Ошибка подтверждения заказа: ...".

**Описание вспомогательных функций**

**generateUniqueBarcode**

Функция для генерации уникального штрих-кода. Возвращает строку с barcode, которая гарантированно уникальна в базе данных.

**mockBookingAPI**

Эмулятор внешнего API для бронирования заказа. Принимает массив данных и возвращает случайный ответ:

* Успешный ответ: ['message' => 'order successfully booked']
* Ошибки: ['error' => 'barcode already exists']

**mockApprovalAPI**

Эмулятор внешнего API для подтверждения заказа. Принимает массив данных и возвращает случайный ответ:

* Успешный ответ: ['message' => 'order successfully approved']
* Возможные ошибки: ['error' => 'event cancelled'], ['error' => 'no tickets'], ['error' => 'no seats'], ['error' => 'fan removed']