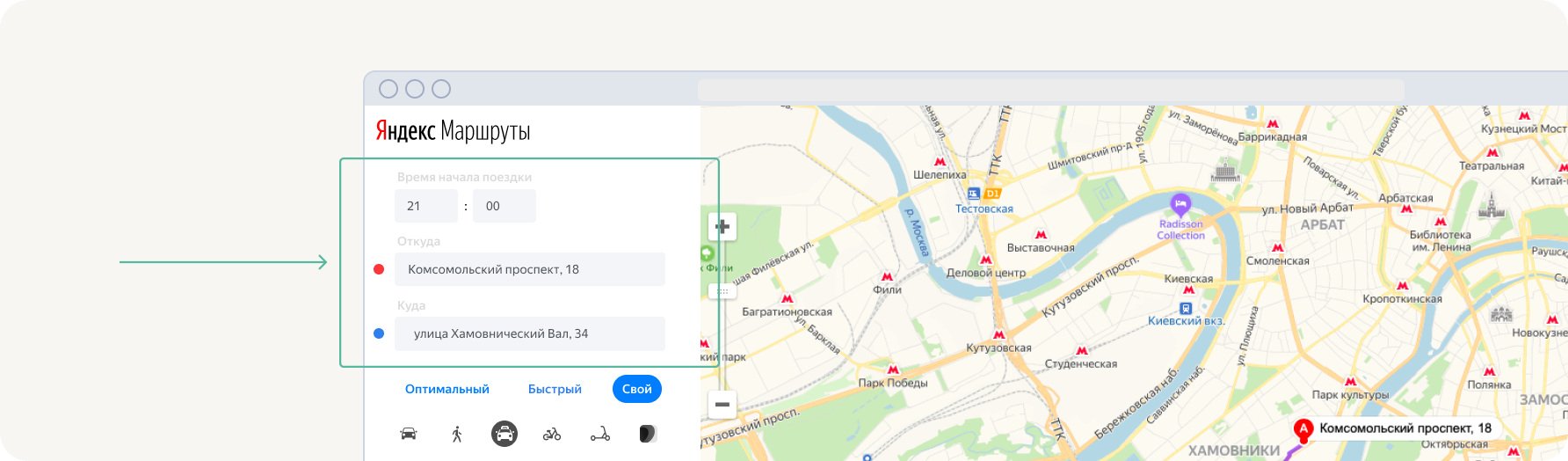
### **Часть 1**

**1. Визуализируй требования**

Проанализируй требования к сервису Яндекс.Маршруты и дорисуй mindmap. Убедись, что тебе удалось декомпозировать все требования. С mindmap можно сверяться, пока ты будешь выполнять другие части задания — так не придётся постоянно обращаться к требованиям.

**2. Выдели классы эквивалентности и граничные значения для полей ввода**

Тебе нужны поля «Время начала поездки», «Откуда», «Куда».



Выдели классы эквивалентности.

Определи граничные значения каждого класса, если применимо.

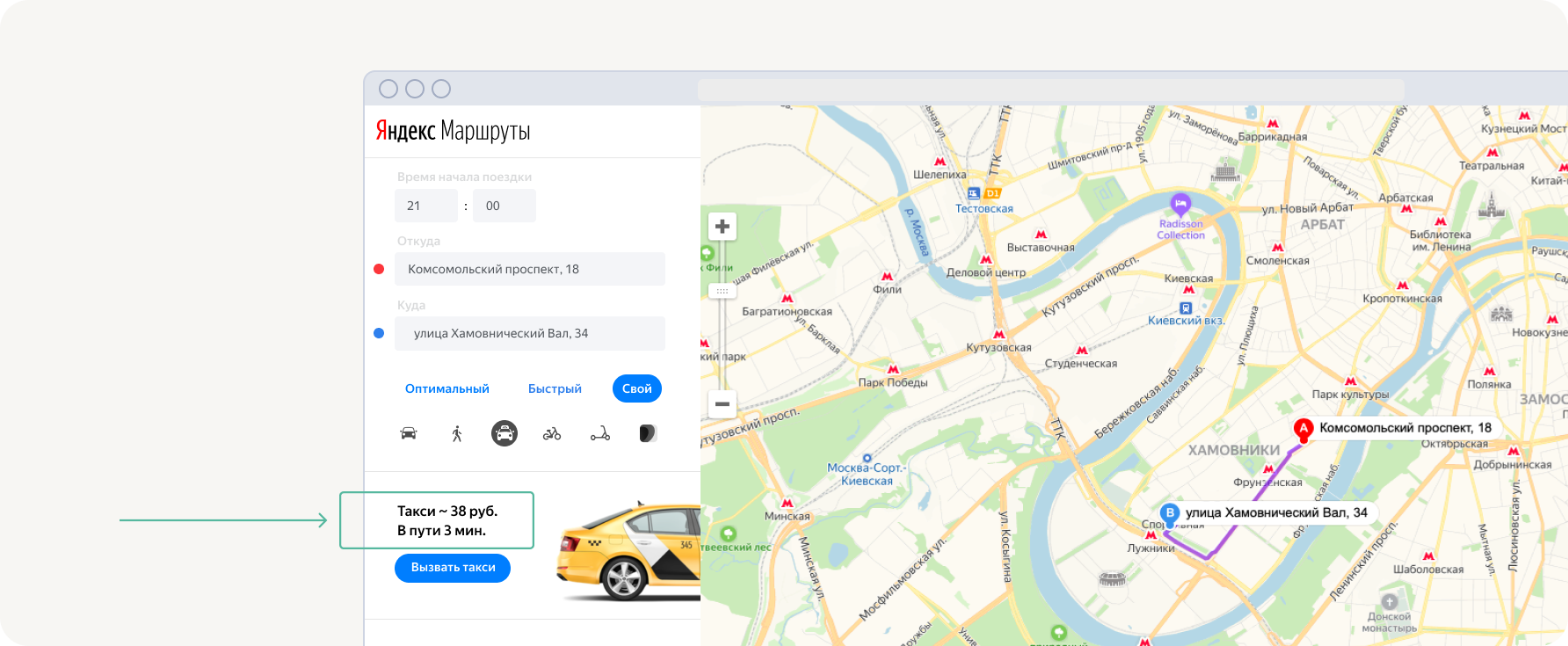
Выбери тестовые значения, которые проверят каждый класс; и его границы, если они есть.

Не забудь проверить негативные сценарии.

### **Часть 2**

**Спроектируй тесты для расчёта стоимости и времени**

Приложение выводит стоимость и время поездки. В расчётах могут быть ошибки, поэтому важно проверить, что стоимость и время поездки рассчитываются правильно.



Как это сделать:

Выбери один вид транспорта для тестирования: собственный автомобиль, каршеринг или такси.

Определи, какие требования описывают логику расчёта стоимости и времени **выбранного транспорта**. Здесь могут помочь в mindmap и/или требования.

Изучи логику расчёта стоимости и времени поездки. Запиши их в виде формул. Они пригодятся чуть позже — когда ты будешь рассчитывать тестовые значения.

Чтобы рассчитать тестовые значения по формулам, тебе понадобятся значения скорости движения. Эти данные есть в таблице, которая показывает зависимость скорости от времени начала поездки. Она находится в требованиях. Визуализируй логику выбора скорости в виде блок-схемы, чтобы не заглядывать в требования каждый раз, когда нужны эти данные.

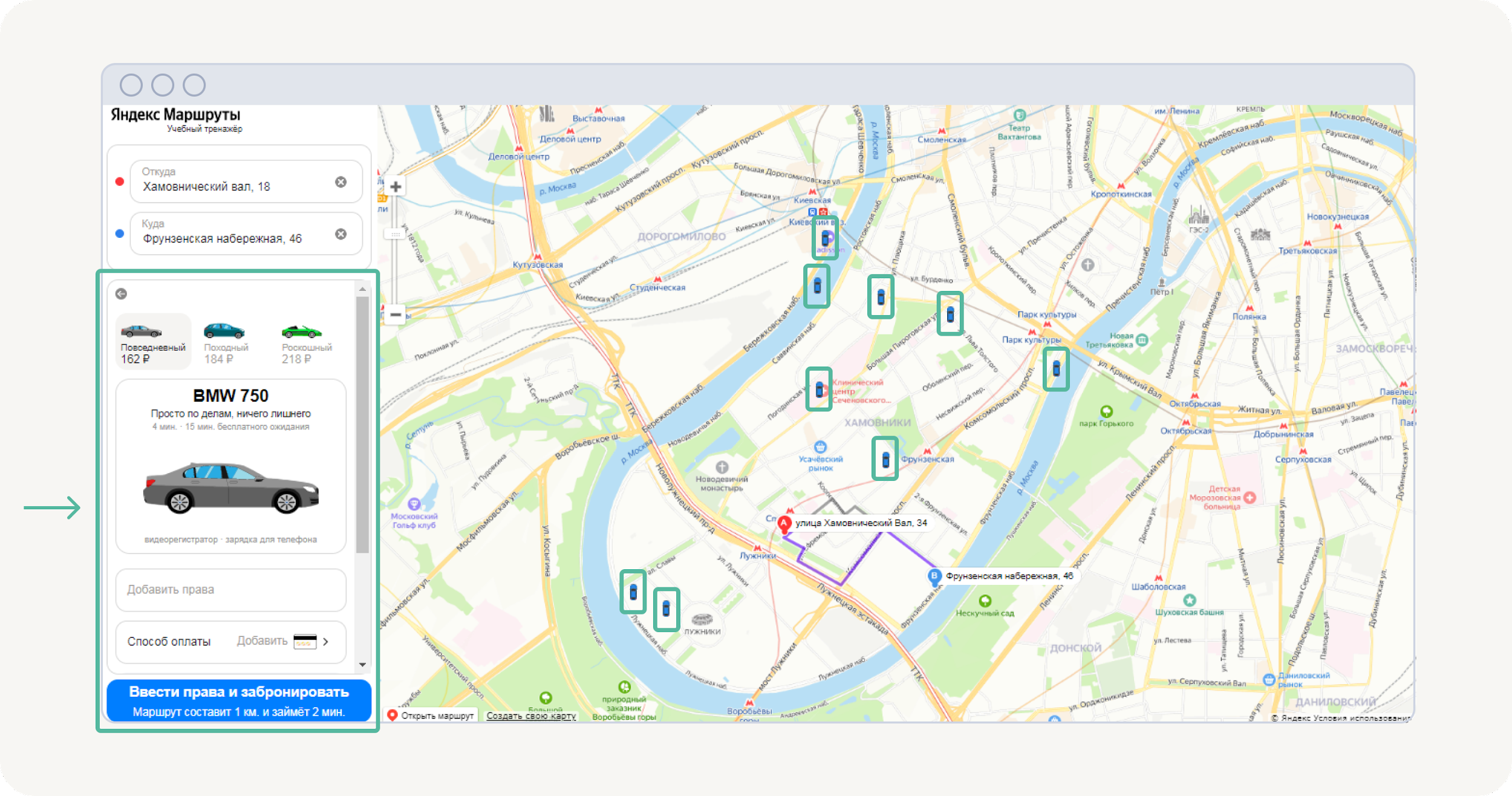
Когда ты выведешь формулы, ты заметишь, что результаты расчётов стоимости и времени зависят от других параметров. Тебе нужно проверить, что логика расчётов учитывает эти зависимости — это нужно сделать на разных данных. Поэтому необходимо определить классы эквивалентности для этих параметров.

Выбери тестовые значения, которые проверят каждый класс и границы, если они есть.

Напиши тест-кейсы на основе тестовых значений внутри классов эквивалентности. Тест-кейсы должны проверять корректность логики расчёта времени и стоимости поездки.

**Часть 3**

**Подготовь тестовую документацию, чтобы проверить вёрстку формы бронирования и навигационной карты.**

****

Чтобы это сделать:

Внимательно изучи макеты каршеринга в Figma — вкладка «Marshruti car sharing».

Обрати внимание на блок «Требования к заказу». В макетах изображен не каршеринг, а такси: это вкладка «Marshruti taxi» → UI kit. Но принцип работы везде одинаковый. Отличаются только тарифы и текст.

Подробнее о работе панели «Требования к заказу» в каршеринге можно прочитать в текстовых требованиях.

Составь чек-лист, по которому будешь тестировать **вёрстку формы бронирования и элементов на навигационной карте: это иконки автомобилей и действия с ними.**.

Выбери только один тариф. Например, «Роскошный».

Валидацию полей, а также вёрстку окон «Добавление прав», «Способ оплаты», «Добавление карты» тестировать **не нужно**.

**2. Подготовь тестовую документацию, чтобы проверить логику работы.**

Проанализируй требования к функциональности каршеринга.

Составь чек-лист, по которому будешь проверять функциональность окон «Способ оплаты» и «Добавление карты». Здесь пригодятся соответствующие пункты требований: «Поле “Способ оплаты”» и «Окно “Добавление карты”».

Когда будешь составлять чек-лист, используй технику разбиения на классы эквивалентности и выделения граничных значений.

Не забудь про **позитивные и негативные проверки**.

Подготовь тест-кейсы:

На логику работы кнопки «Забронировать» — см. пункт требований «Кнопка “Забронировать”».

На логику функциональности бронирования — см. пункт требований «Бронь машины».

**3. Протестируй приложение и заведи баг-репорты**

Протестируй функциональность каршеринга по чек-листам и тест-кейсам, которые тебе удалось спроектировать ранее. Времени на проверку осталось мало, поэтому используй только две конфигурации окружения:

Яндекс.Браузер при разрешении экрана 800x600;

Firefox при разрешении экрана 1920x1080.

Тестирование вёрстки нужно провести в **обоих** окружениях. Логику приложения достаточно проверить в **одном** окружении.