Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Ульяновский государственный технический университет»
Кафедра «Вычислительная техника»

Лабораторная работа №4

Вариант 12.

Kazan Express - Оформление доставки

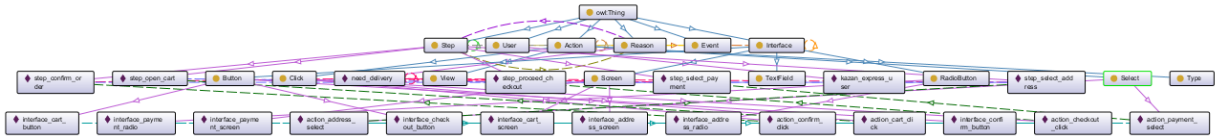
Выполнил:
студент группы: ИВТИИбд-13
Кузнецов А. А.

Проверил:
преподаватель кафедры «ВТ»
Хайруллин И. Д.

Ульяновск
2025

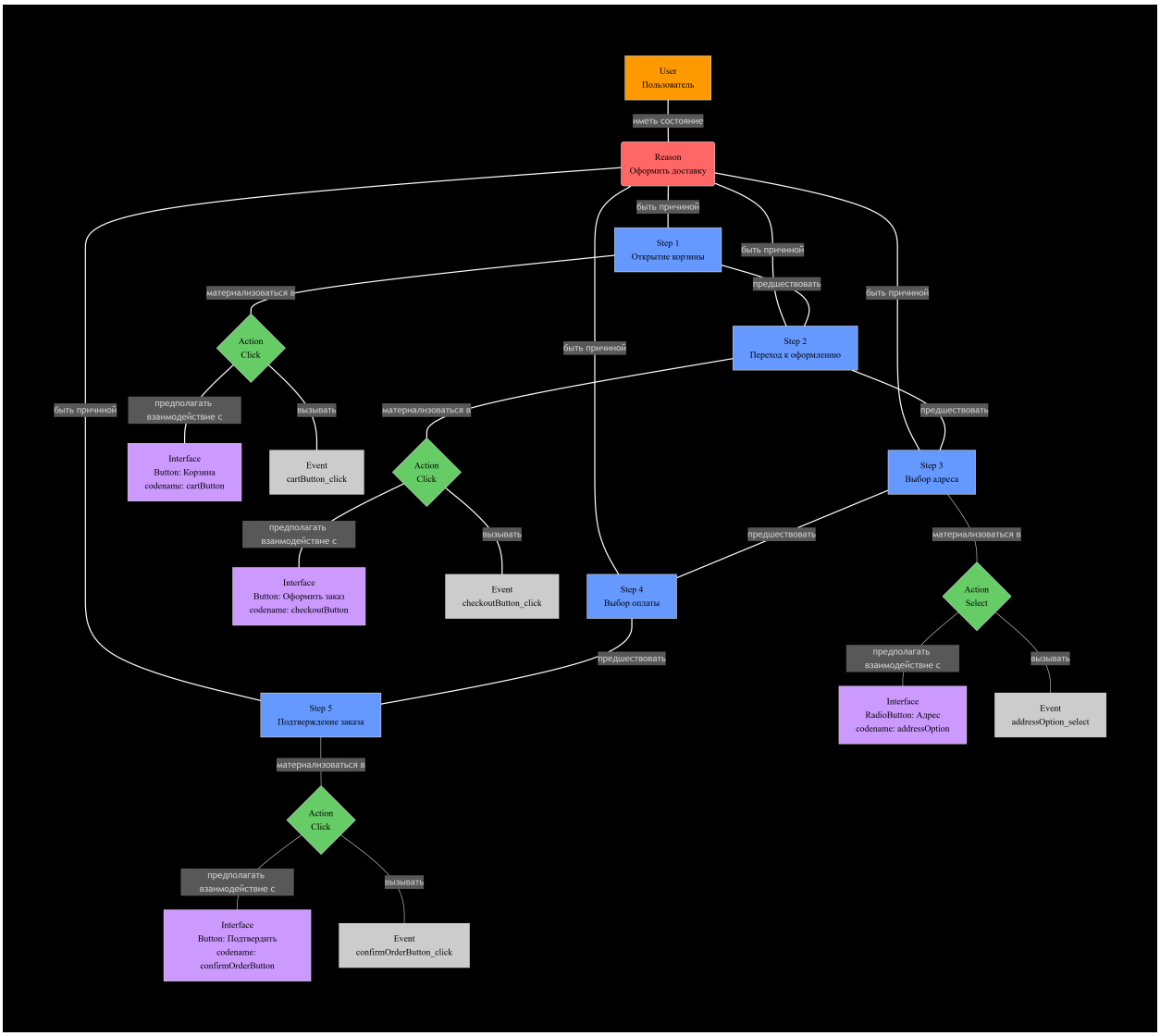
1 До «Reasoner»

2 После «Reasoner»



3 Графическое представление модели

Сделал в paint



Описание онтологической модели и задействованных правил

1.1. Общее описание

Разработанная онтологическая модель описывает сценарий оформления доставки в мобильном приложении "Kazan Express". Модель построена на основе онтологического подхода и включает все необходимые сущности для описания пользовательского взаимодействия с системой.

1.2. Классы модели

Основные классы онтологии

- User (Пользователь) - представляет пользователя мобильного приложения

- Reason (Причина) - описывает цель использования сервиса ("Оформить доставку")
- Step (Шаг - этапы сценария оформления заказа
- Action (Действие - конкретные действия пользователя или системы
- Interface (Интерфейс - элементы пользовательского интерфейса
- Event (Событие - логируемые события для аналитики

Специализированные подклассы

- Action подклассы
  - Click - нажатие на элементы интерфейса
  - Select - выбор из вариантов
  - View - просмотр экранов
  - Type - ввод текста

- Interface подклассы
  - Button - кнопки интерфейса
  - Screen - экраны приложения
  - RadioButton - радиокнопки для выбора
  - TextField - поля для ввода текста

1.3. Отношения между классами

Модель включает следующие объектные свойства (отношения):

- has_state - связывает User с Reason (пользователь имеет состояние)
- is_reason_for - связывает Reason с Step (причина порождает шаги)
- precedes - определяет последовательность Step (шаги следуют друг за другом)
- materializes_into - связывает Step с Action (шаг материализуется в действия)
- interacts_with - связывает Action с Interface (действие выполняется через интерфейс)
- is_part_of- определяет иерархию Interface (элементы входят в состав экранов)
- triggers - связывает Action с Event (действие порождает событие)

1.4. Свойства данных

- name - текстовое название для всех классов
- codename - формальное название для элементов интерфейса (используется в коде)

1.5. Экземпляры модели

Пользователь и причина:
- kazan_express_user - пользователь приложения
- need_delivery - цель "Оформить доставку"

Шаги сценария:
1. step_open_cart - Открытие корзины

2. step_proceed_checkout - Переход к оформлению
3. step_select_address - Выбор адреса доставки
4. step_select_payment - Выбор способа оплаты
5. step_confirm_order - Подтверждение заказа

Действия:
- action_cart_click - Нажатие на корзину (Click)
- action_checkout_click - Нажатие на оформление заказа (Click)
- action_address_select - Выбор адреса доставки (Select)
- action_payment_select - Выбор способа оплаты (Select)
- action_confirm_click - Нажатие на подтверждение заказа (Click)

Элементы интерфейса:
- interface_cart_button - Кнопка "Корзина" (codename: cartButton)
- interface_checkout_button - Кнопка "Оформить заказ" (codename: checkoutButton)
- interface_address_radio - Радиокнопка выбора адреса (codename: addressOption)
- interface_payment_radio - Радиокнопка выбора оплаты (codename: paymentOption)
- interface_confirm_button - Кнопка "Подтвердить заказ" (codename: confirmOrderButton)

2. Описание SWRL правил

2.1. Общее назначение правил

SWRL правила реализованы для автоматического создания связей между действиями и событиями. Правила обеспечивают автоматическое наполнение базы данных информацией о событиях на основе взаимодействия пользователя с интерфейсом.

2.2. Формат именования событий

События именуются по шаблону: `[codename]_[тип_действия]`
- codename - формальное название элемента интерфейса
- тип_действия - click, select, view, type

2.3. Реализованные правила

Правило 1: Для действия с корзиной**
swrl
```
Step(?s) ^ •_materializes_into(?s, action_cart_click) ^
•_interacts_with(action_cart_click, ?i) ^
codename(?i, "cartButton") ^
Event(?e) ^ name(?e, "cartButton_click")
->
•_triggers(action_cart_click, ?e)
```

Логика правила: Если шаг материализуется в действие нажатия на корзину, которое взаимодействует с интерфейсным элементом с codename "cartButton", и существует событие с именем "cartButton_click", то действие порождает это событие.

Правило 2: Для оформления заказа**
swrl
Step(?s) ^ •_materializes_into(?s, action_checkout_click) ^
•_interacts_with(action_checkout_click, ?i) ^
codename(?i, "checkoutButton") ^
Event(?e) ^ name(?e, "checkoutButton_click")
->
•_triggers(action_checkout_click, ?e)

Правило 3: Для выбора адреса
swrl
Step(?s) ^ •_materializes_into(?s, action_address_select) ^
•_interacts_with(action_address_select, ?i) ^
codename(?i, "addressOption") ^
Event(?e) ^ name(?e, "addressOption_select")
->
•_triggers(action_address_select, ?e)

Правило 4: Для выбора оплаты
swrl
Step(?s) ^ •_materializes_into(?s, action_payment_select) ^
•_interacts_with(action_payment_select, ?i) ^
codename(?i, "paymentOption") ^
Event(?e) ^ name(?e, "paymentOption_select")
->
•_triggers(action_payment_select, ?e)

Правило 5: Для подтверждения заказа
swrl
Step(?s) ^ •_materializes_into(?s, action_confirm_click) ^
•_interacts_with(action_confirm_click, ?i) ^
codename(?i, "confirmOrderButton") ^
Event(?e) ^ name(?e, "confirmOrderButton_click")
->
•_triggers(action_confirm_click, ?e)

2.4. Результаты работы правил

После выполнения правил устанавливаются следующие связи:
- action_cart_click → triggers → cartButton_click
- action_checkout_click → triggers → checkoutButton_click
- action_address_select → triggers → addressOption_select
- action_payment_select → triggers → paymentOption_select
- action_confirm_click → triggers → confirmOrderButton_click