AI-помощник туриста — презентация решения (GORKYCODE 2025)

Команда МыВместе

1) Вводные: почему идея важна, ЦА и её боли

Целевая аудитория

- Гости города с 1–3 часами «окна» между делами.
- Местные, которым хочется тематической прогулки: стрит-арт, панорамы, кофейни по пути.
- Малые компании без авто — **пешие маршруты** в радиусе 1–5 км.

Боли

- Слишком много вариантов, непонятно **куда успею дойти** за отведённое время.
- Подборки нападобии «Топ-10 мест» не учитывают **интересы** и **стартовую точку**.
- Карты показывают метки, но не дают **готового оптимального порядка обхода** и **таймингов**.
- Текстовые описания раздутые, часто с выдумками.

Как решаем

- Индивидуальный список точек по интересам → **маршрут, уложенный во время** → лаконичные причины «почему сюда» — всё в одном ответе.
- Мы **подбираем точки по смысловой близости к интересам пользователя и по их географической близости к старту**, а затем строим **реальный пешеходный маршрут** по уличной топологии.
- Семантический поиск (модель `intfloat/multilingual-e5-large` + Qdrant) находит релевантные места по описаниям.
- Реранжирование смешивает **семантику** и **дистанцию до пользователя**: финальный счёт `final = α·sem + (1−α)·exp(−d/τ)`.
- Маршрут и время в пути считает **поднятый OSRM** (профиль `foot`), который учитывает **топологию пешеходных дорог** и реальную проходимость.
- **LLM используется только для персонального рассказа о точках** и объяснения «почему туда» — он **не участвует** в расчёте/планировании маршрута. Это снижает риск галлюцинаций и делает систему более быстрой и интерпретируемой(можно полагаться на скоры семантики и гео-ранжирования). Применение сверхкрупных моделей с большим кол-вом обучающихся параметров для самого планирования могло бы снизить риск, но это **неоптимально** по стоимости/задержке и всё равно уступает специализированным алгоритмам маршрутизации.

2) Объяснение решения: технологии, подходы, ключевые особенности и почему так

Архитектура (конвейер)

1. **Ввод пользователя** в Telegram (aiogram FSM): интересы → часы → способ указания позиции (гео, адрес, координаты).
2. **Геокодирование** адреса через Nominatim (если нужно).
3. **Поиск кандидатов**: Qdrant + `intfloat/multilingual-e5-large` (SentenceTransformers).
4. **Гео-реранжирование**: смешиваем семантику и расстояние до старта.
5. **Подбор количества точек и порядка обхода** под временной бюджет:
   - матрица длительностей/дистанций через **OSRM**,
   - стартовая перестановка **Nearest Neighbor** → улучшение **2-opt**,
   - считаем ходьбу + «осмотры» и **останавливаемся ровно там, где ещё укладываемся**.
6. **Вывод**: интерактивная карта (Folium) + компактный текст-гид (Gemini).

Данные и индекс

- Источник: обогащенный датасет культурных объектов («Мой Нижний Новгород») + поля координат.
- **Предобработка и хранение в векторной бд Qdrant** :
  - `text_for_embedding()` : **категория + синонимы + заголовок + очищенное описание** → строка для эмбеддинга.
  - Модель эмбеддингов: `intfloat/multilingual-e5-large` (отличное качество на русском).
  - Храним в **Qdrant** (COSINE, нормализованные векторы), индексы по category/title/address.

Почему так: E5-семейство стабильно на запросах в стиле «query:… / passage:…», Qdrant даёт быстрый ANN-поиск и фильтрацию по категориям.

Семантика + гео: как считаем «актуальность» точки для пользователя

В `geo_search.semantic_proximity_rerank()`:

- **Семантический скор**: `sem01 = (cos_sim + 1) / 2` (значение в [0;1]).
- **Гео-скор**: `geo = exp(-distance_km / τ)`, где `τ = geo_tau_km` (по умолчанию ~1.2 км).
- **Итог**: `score = α * sem01 + (1 - α) * geo` (в приложении используется `α=0.8` — семантика главнее).
  Порог «очень далеко» — `hard_drop_km=30` (срез дальних выбросов).

Почему так: Пользовательские интересы — главный сигнал, но для прогулки важна **пешая достижимость**. Экспонента по расстоянию даёт «главный» штраф, не рубит по радиусу.

«Основные» vs «Дополнительные» точки

- **Основные**: чистый **топ-k по семантике** (`semantic_topk`, `main_semantic_k=5`).
  Если «основная» в радиусе `pin_radius_km=8`, её **продвигаем** вверх итогового списка.
- **Дополнительные**: хорошие по финальному скору, но не попали в «основные» — интересные места, на которые можно обратить внимание по пути.
- **Длительности осмотра**: `15 мин` для основных и `3 мин` для дополнительных.

Почему так: Для разнообразия и полноты маршрута концентрируемся на основных точках, но оставляем пространство для осмотра достаточно подходящих доп. точкек по пути.

Укладка во временной бюджет

1. Тянем **OSRM /table** для матрицы времен/дистанций и **/route** для полилиний.
2. Строим путь: NN → 2-opt (фиксируем порядок `fixed_order`).
3. **Итерируем N** (сколько брать точек): считаем
   - **ходьбу** (OSRM-время, умножаем на `pace_scale`),
   - **осмотры** (15/3 мин, кроме финальной точки),
   - **останавливаемся на максимальном N**, которое **≤ бюджет**.
4. Если «не влезло» → понятная подсказка: **сколько минут добавить**, чтобы дойти до ближайшей «основной».

Параметры по умолчанию и в приложении:

- `pace_scale=1.67` → целевая скорость ≈ `5 / 1.67 ≈ 3 км/ч` (реалистично для прогулки с остановками).
- `roundtrip=False` (не замыкаем на старт).

Почему так: OSRM даёт **реалистичные тайминги пешком**, а NN+2-opt — быстрый и стабильный порядок без тяжёлого TSP. Итерация по N гарантирует, что выдача **не сорвёт бюджет** времени.

Текст-гид

`gorky_guide.get_reason()`:

- Модель **Gemini 2.5-flash** генерирует **строго JSON** по заданной схеме.
- В prompt запрещены домыслы: используем **только** факты из полей объекта.
- Рендер в чат (`render_guide_text`) — компактный, без Markdown-шума.

Почему так: LLM используется **только для краткого объяснения**, а не для подбора — это снижает риск галлюцинаций и делает ответы воспроизводимыми.

Телеграм-UX

- Чёткий **мастер из 3 шагов** (aiogram FSM) + легкий рестарт.
- Способы указания локации: гео-позиция, адрес (Nominatim), координаты.
- Чёткие **ошибки и подсказки**: `time_budget_too_small`, `no_candidates`, «добавьте ~N минут» и т. п.
- Карта сохраняется как HTML и отдается по `PUBLIC_BASE_URL` с кнопкой «Открыть маршрут».

Гиперпараметры (что крутится «снаружи» и зачем)

Параметр	Где	Значение по умолчанию / пример	Смысл
`alpha`	`semantic_proximity_rerank` → `plan_route_under_budget`	0.8	Баланс семантики и близости. Больше → важнее интересы, меньше → важнее расстояние.
`geo_tau_km`	реранж	1.2	«Длина затухания» штрафа по расстоянию.
`fetch_k`	поиск кандидатов	50–80	Сколько брать кандидатов из Qdrant до реранжа/сборки маршрута.
`main_semantic_k`	основные	5	Сколько точек считаем «главными» по чистой семантике.
`pin_radius_km`	промо «основных»	8.0	Радиус, где «главные» поднимаются выше в общем списке.
`hard_drop_km`	реранж	30	Жёсткий срез «слишком далёких» объектов.
`pace_scale`	тайминги	1.0–1.67	Масштаб скорости пешком. 1.67 ≈ 3 км/ч.
`roundtrip`	маршрут	False	Возвращаться ли к старту.
`dwell_minutes_per_node`	осмотры	15/3	Время осмотров: основные/дополнительные.

3) Почему этот сервис «лучший» для пользователя

1. **Гарантия по времени.** Мы не просто выдаём «список мест» — выдаём **проходной порядок**, который укладывается в заданные часы, с прозрачной легендой и таймингами.
2. **Объяснимость.** Подбор — на эмбеддингах и геометрии, а LLM только «объясняет» — меньше галлюцинаций, больше доверия.
3. **Реалзим пешком.** OSRM учитывает топологию уличной сети; `pace_scale` калибрует под «городскую прогулку» ~ 3 км/ч.
4. **Баланс интересов и расстояния.** Формула `α·семантика + (1−α)·гео` даёт ровно то, что хочет пешеход: интересно и близко.
5. **Понятные подсказки, если маршрут не влезает по времени.** Вместо «ничего не нашли» — даём, **сколько минут добавить**, и ближайшую достойную «основную» точку инттереса.
6. **Лёгкий старт и нулевой барьер.** Работает прямо в Telegram, рассказ об интересах в свободном формате: от ключевых слов до полных предложений. Несколько способов задачи текущей позиции пользователя: гео-метка, адресс, координаты. Интерактивная карта открывается одной кнопкой в telegram mini app.
7. **Масштабируемость.** Qdrant, SentenceTransformers и OSRM легко расширяются на другие типы активностей: все что нужно сделать это дополнить датасет.