На рынке жилья сегодня очень много предложений и все они разные. И для того что бы сделать верный выбор необходимы инструменты позволяющие быстро искать и сравнивать их. На сегодняшний день самыми крупными сайтами по продаже жилья в РФ являются cian.ru и domofond.ru. И они оба имеют возможность поиска по карте. Но я не считаю, что они используют потенциал карты полностью. Рассмотрим скриншоты этих сайтов. Все что пользователь может понять это наличие предложения в определенной точке. На cian.ru видно только количество предложений. На domofond же видно ещё и является ли предложение вторичным или первичным. И это все на что способны данные ничтожные сайты.

Карту можно сделать более информативной раскрашивая дома в цвет, зависящий от цены за квадратный метр, и показать пользователю транспортную доступность раскрашиваю дороги в цвет зависящий от их удаленности от точки отсчета. Градиенты приведены на слайде. Благодаря естественной ассоциации красного как с чем-то более дорогим, а синего как с чем-то дешевым. Также в тепловизоре красным цветом показываются горячие области, а синим наоборот холодные. Поэтому мы можем сопоставить максимальные цены с красным, а минимальные с синим. Тот же принцип применим к дальности.

Для реализации подобного функционала необходимы геоданные и цены. Геоданные нужны для отрисовки и расчета дальности. Цены для того что бы было что визуализировать. Геоданные и цены я брал для области приблизительно ограниченной бетонным кольцом.

В качестве сервера мною был выбран nginx т.к. он очень популярен и используется крупнейшими интернет компаниями. Java была выбрана в качестве языка программирования из-за того что я неплохо владею ей, и её очень часто применяют в сфере написания серверов.

Все запросы сначала попадают на сервер nginx, дальше в зависимости от типа запроса либо nginx отдаёт уже готовую картинку карты, либо перенаправляет запрос на сервер java. Его принимает Netty, дальше он передаётся в PathRouter, который согласно url передаёт запрос нужному handler.

Для отрисовки и поиска по карте необходимо организовать данные на плоскости. Для этого существует несколько способов, но я выбрал квадродерево, как самый простой в реализации. У квадродрева у каждого узла 4 потомка, которые делят прямоугольник родителя на 4 равных части. Разделение узла происходит, когда количество элементов в нем превышает некий порог. Благодаря этому нам никогда не приходится искать перебором в количестве элементов большем чем порог разделения.

Из-за плохой стандартизации данных имеются проблемы с маршрутами, остановки не являются частью маршрута. Пришлось делать остановки частью ближайшего к ним ребра. А после соединить остановку с ближайшими вершинами. После этого маршруты общественного транспорта становятся частью дорожного графа.

В начале из-за простоты реализации использовался расчет на основе поиска в глубину, но из-за его плохой производительности я переписал расчет на основе поиска в ширину. Их сравнение приведено на слайде.