**ОБЩЕЕ ПРОЕКТИРОВАНИЕ**

В настоящем проекте выделяется несколько логических частей, которые можно выделить как по технологиям, так и по бизнес-задачам. На текущем этапе они включают: десктопное и мобильное приложения, сайт с информацией о проекте и возможностью загрузки приложений, главный сервер, VPN-сервера.

Изучение результатов тестирований производительности различных VPN протоколов показало, что Wireguard обеспечивает наибольшую скорость и наименьший процентный объем служебного траффика. Также, у разработчика настоящего проекта имеется опыт работы с данном протоколом, поэтому он выбран как приоритетный для использования.

Десктопное и мобильные приложения не являются главным приоритетом разработки с точки зрения качества. Они должны предоставлять графический интерфейс, реализующий взаимодействия с используемым VPN-протоколом. Для разработки десктопного приложения предлагается использовать WPF, мобильного – Kotlin. Изучение документации­[1] Wireguard показало отсутствие существенных преград для его встраивания в разрабатываемые приложения, также был изучен конкретный пример встраивания WG в C# приложение.

Сайт должен представлять интерфейс для функций регистрации и управления аккаунтов с помощью API главного сервера и возможность загружать клиентские приложения. Для разработки предлагается использовать ReactTS.

Главный сервер должен предоставлять возможность регистрации и управления аккаунтом, просмотра текущих устройств, скачивания клиентcких приложений, подключения к VPN-серверам. Для разработки предлагается использовать стек PostgreSQL+ASP.

Все серверные службы предлагается развертывать с применением docker-compose. Даже в отсутствии необходимости оркестрации, коя, на текущем этапе, видится только у главного сервера, docker-compose не снижает производительность, однако может упрощать развертывание относительно отдельных контейнеров.

Сами VPN-сервера должны представлять собой легкие в развертывании docker-compose сборки, работающие c применением протокола Wireguard и ASP в качестве приложения, управляющего конфигурацией. Изучение документации[2] docker показало, что разумным вариантом является запуск двух приложений в одном контейнере, нежели их разделение, т.к. в этом случае возникают проблемы с получением доступа из одного контейнера в другой, а также нарушается в целом идея docker-контейнеризации, которая подразумевает взаимодействие с контейнерами по веб-протоколам (в т.ч. HTTP, RPC). Проблема состоит в том, что Wireguard не отслеживает обновление файла конфигурации. Таким образом, если ASP и может изменить файл конфигурации Wireguard (путем, например, использования docker volumes), то это не привет к обнаружению и применению этого изменения. Наивным решением могла бы быть регулярная перезагрузка файла конфигурации, например с применением cronjob, однако это однозначно более медленно, нежели стандартный способ добавления пиров «на лету», т.е. через выполнение команды ‘wg set wg0 peer KEY allowed-ips IP’[2].