Требования к клиентской машине

От клиентской машины требуется веб-браузер Google Chrome или любой другой chromium-based браузер не ниже 51 версии.

Требования к серверной машине

Для работы тонкого клиента необходим веб-сервер с функцией Reverse-Proxy.
Это может быть NGINX или другие, с поддержкой Reverse-Proxy.
Мы рассматриваем поставку для NGINX.

Windows

Аппаратное обеспечение сервера

- Процессор x86-совместимый, 64-разрядный (AMD, Intel). 4 ядра с частотой от 2ГГц.
- Оперативная память 16Гб.
- Дисковая подсистема от 50Гб.
- Канал связи от 100Мбит/с, при кол-ве пользователей >50 рекомендуется от 1000Мбит/с.

Программное обеспечение сервера

- Для управления веб-сервером требуется веб-сервер nginx (не ниже 1.22.1).
- Для настройки сервисов веб-сервера потребуется NSSM - the Non-Sucking Service Manager.
- .Net 6.0 runtime
- Windows Server версии 2012 и выше
- Региональные настройки – язык Русский и расположение – русский.

Linux

Аппаратное обеспечение сервера

- Процессор x86-совместимый, 64-разрядный (AMD, Intel). 4 ядра с частотой от 2ГГц.
- Оперативная память 16Гб.
- Дисковая подсистема от 50Гб.
- Канал связи от 100Мбит/с, при кол-ве пользователей >50 рекомендуется от 1000Мбит/с.

Программное обеспечение сервера

- Веб-сервер nginx.
- .Net 6.0 runtime.
- Ubuntu 20.04.5 и выше.

Поставка

В поставке участвуют три или четыре каталога:

FinistDigitalBank.Report.Server.Worker – Сервер-приложение.
FinistDigitalBank.Report.WebClient.Worker (далее – веб-приложение): Сервер-приложение для тонкого клиента и Клиент-приложение для сервера.
FinistDigitalBank.Report.WDS (далее – тонкий клиент): Файлы тонкого клиента.
EventManager.UserArm (далее – толстый клиент): Файлы толстого клиента. Только для Windows.

Если ПО ставится впервые, то так же присутствует файл базы данных.

Разворачиваем приложения на сервере

Windows

Копируем каталоги, участвующие в поставке. После, можно переходить к настройке приложений.

Linux

Нам понадобится Far Manager и клиент SSH, например Putty.
Можно использовать и другие клиенты с поддержкой sftp и командной строки.

В примере используется Far Manager.

Навигация в Far

Alt + F1 / F2 – переключение отображаемой директории в окнах, где F1 – левое, а F2 – правое.
Tab – переключение между окнами.
Стрелки на клавиатуре – перемещение в окне.
F5 – копирование файлов.
F7 – создание нового каталога в открытом окне.
F8 – удаление файлов/каталогов.
Insert – выделение файлов/каталогов.

Перенос файлов при помощи FAR

При помощи Far'а создаём новое подключение командами Alt + F1 (изменить левое окно), переключаемся на NetBox

В открывшейся вкладке создаём новое соединение при помощи Shift + F4

Заполняем поля полученными данными. Нажимаем Ok

Появится предупреждение, с вопросом – «точно хотим сохранить пароль?». Сохраняем. Далее, откроется окно, где мы можем дать удобное наименование новому подключению:

После, в списке появится новая запись.

Нажатием Enter на записи – открываем новое соединение

После успешного подключение – в левой части экрана будут отображаться каталоги сервера, а справа – каталоги вашего компьютера.

Создаём новый каталог «finist-soft» при помощи клавиши F7

При наличии ошибки Permission denied – необходимо запросить от банка права на создание и копирование файлов для нашего пользователя.
При помощи навигации, в правом окне переходим к каталогу с файлами приложения, выделяем (клавиша insert) и копируем их при помощи клавиши F5, нажимаем Copy

После завершения копирования, переходим к настройке приложений.

Редактирование в FAR

Для изменения файлов конфигурации и настройки приложений необходимо, что бы пользователь обладал правами на редактирование файлов.
Само редактирование происходит при помощи команды Edit (клавиша F4)

После изменений – необходимо выполнить команды Save (клавиша F2).
Или при закрытии изменённого файла при помощи команды Quit (клавиша F10)

Настройка базы данных

PostgreSQL

DBeaver

Для этого способа необходимо, чтобы сервер базы данных уже был настроен и база-пустышка, которую будем восстанавливать уже существует.
Если этого сделано не было, то подключение и настройка БД через DBeaver невозможна.
Воспользуйтесь настройкой по SSH
Рассмотрим развёртку базы при помощи бесплатного ПО DBeaver Community.

Открываем DBeaver, создаём новое подключением нажатием на «вилку», выбираем PostgreSQL двойным нажатием.

Откроется окно с настройками подключения
В данном окне необходимо ввести данные сервера БД.
Host – адрес сервера БД.
Database – база данных, к которой мы осуществляем подключение.
Группа **Authentication:**
Username – пользователь, под которым будет осуществляться управление базой. Он должен иметь права на управление базой.
Password – пароль пользователя БД.

Заполняем все перечисленные поля и нажимаем «ОК»

В случае успешной настройки, в навигаторе мы увидим наше подключение. Разворачиваем его.

Если всё успешно, то возле значка postgresql появится зелёная «галка».
Далее, разворачиваем список Databases и находим нужную базу данных

Нажимаем по строке eventmanager ПКМ, переходим в Tools, затем нажимаем Restore

Откроется окно восстановления базы

Нас интересует строчка **Backup file**
Указываем полный путь или нажимаем на значок «каталог»
Откроется диалоговое окно с выбором файла бд, необходимо сменить расширение файла на * (любой)

После этого выбрать файл

И нажать кнопку Start

Запустится процесс восстановления базы из файла. После завершения можно приступать к настройке сервера.

SSH

Подключаемся по SSH до сервера БД.
SSH-клиент встроен в windows и linux. В win необходимо открыть PowerShell и исполнить команду:
`ssh **user**@**remoteAddress**`
В случае, если банк использует специфичный порт для подключения, необходимо добавить аргумент **-P**
`ssh -P **portNumber** **user**@**remoteAddress**`
**красные поля* – *значения, которые нужно заполнить данными от банка.*

После успешного подключения, система запросит пароль. При вводе пароля система будет вести себя так, будто ничего не происходит, но это не так. Ввод пароля осуществляется, но никак визуально не отображается.
После ввода, нажмите Enter, если пароль введён верно, то выведется приветственный текст и появится возможность исполнять команды на сервере
Перед развёрткой базы необходимо проверить, установлены ли на сервере подходящие локали, это делается командой.
Если ее нет, то выполняем команду `sudo locale-gen ru_RU.UTF-8`

И обновляем данные о loale командой `sudo update-locale`
Находим сервис БД командой `systemctl | grep postgres`
Нас интересует main сервис. Перезапускаем его командой `systemctl restart postresql@14-main.service`
После перезапуска, переключаем пользователя на postgres. Если такого пользователя нет, то убедитесь, что сервер БД развёрнут корректно. Документация от DigitalOcean.

Переходим в консоль управления psql и создаём новый инстанс базы данных командой

```
CREATE DATABASE eventmanager WITH
       TEMPLATE template0
       OWNER = postgres
       ENCODING = 'UTF8'
       LC_COLLATE = 'ru_RU.UTF-8'
       LC_CTYPE = 'ru_RU.UTF-8'
       TABLESPACE = pg_default
       CONNECTION LIMIT = -1;
```

Убедимся, что база создалась командой

Необходимо выйти из под учётной записи postgres и переключиться на root.
Далее, необходимо развернуть базу, отправленную вместе с сервисами. Это делается командой
`pg_restore -U postgres -d eventmanager dumpfilepath`
*где **dumpfilepath** является путём до файла бд*
Теперь база развёрнута и необходимо создать пользователя, который будет использован для взаимодействия ПО с базой данных.

Вновь переходим к пользователю postgres и его консоли psql. Подключаемся к стандартной базе postgtresql – template1. Используем команды \connect template1 и создаём роль командой

```
create role gestuser with superuser nocreatedb nocreaterole noinherit login
noreplication nobypassrls password 'Cp12NsSg';
```

Мы создали пользователя gestuser, далее, необходимо раздать ему права доступа до базы данных приложения. Отключаемся от текущей базы командой exit и подключаемся к бд приложения, в данном примере это eventmanager.
Последовательно исполняем следующие команды:

```
grant connect on database eventmanager to gestuser;
grant all privileges on schema dbo to gestuser;
grant all privileges on all tables in schema dbo to gestuser;
grant all privileges on all sequences in schema dbo to gestuser;
```

Данные команды выдадут права пользователю gestuser.

Далее, для созданной базы указываем параметр search_path.

```
SET search_path TO dbo;
```

Данный параметр укажет БД, где в первую очередь необходимо производить поиск таблиц, если в запросе не была указана схема.

Готово. База настроена и готова к использованию.

Настройка приложений

FinistDigitalBank.Report.Server.Worker

Конфигурируется двумя файлами:
appsettings.json
Настройки для работы приложения.
FinistDigitalBank.Report.Server.Worker.dll.config
Набор настроек для сервер-приложения.
В нём указываются строки подключения к БД, настройки для LDAP сервера, почтового сервера.
Общение с веб-приложением происходит посредством gRPC-запросов.

Appsettings.json

Краткое описание файла конфигурации.

```
{
    "AllowedHosts": "*",
    "Kestrel": {
        "EndpointDefaults": {
            "Protocols": "Http2"
        },
        "Endpoints": {
            "Http": {
                "Url": "http://0.0.0.0:5100"
            } //,
            //"Https": {
            //  "Url": "https://0.0.0.0:5101",
            //  // Uncomment certificate section to use specified certificate.
            //  "Certificate": {
            //          "Path": "GrpcCert1.pfx",
            //          "Password": "grpc1"
            //  }
```

```
            //}
        },
        "ServerOptions": { // GEst.Hosting options
            "Limits": {
                "MaxRequestBodySize": 157286400
            }
        }
    },
    "Logging": {
        "LogLevel": {
            "Default": "Information",
            "Microsoft": "Warning",
            "Microsoft.Hosting.Lifetime": "Information"
        },
        "EventLog": {
            "LogName": "Application", // GEst.Hosting options: by default Application
            "SourceName": "MyEvenLogSource", // GEst.Hosting options: by default application name (name of
entry assembly or startup assembly(for web))
            "LogLevel": {
                "Default": "Error",
                "Microsoft.Hosting.Lifetime": "Information",
                "GEst.Hosting.GEstStarterService": "Information",
                "GEst.Hosting.ProgramLogger": "Information"
            }
        }
    },
    "GEst.Hosting": {
        "WindowsService": { // only for Windows services
            "ServiceName": "MyWinService" // by default application name (name of entry assembly or startup
assembly(for web))
        },
        "CORS": { // CORS policies - will be added if specified at least one with nonempty name
            "Policies": [{
                    "Name": "GEst.WebClient.Api", // for api controllers in GEst.WebClient
                    "Origins": ["http://localhost:4205", "http://localhost"]
                }
            ]
        }
    }
}
```

Данный пример можно использовать как начальную конфигурацию для сервера.

AllowedHosts

Указываются адреса, с которых сервер принимает запросы.

Kesterl / EndpointDefaults

Системная секция. Версия протокола общения gRPC.

Kestrel / Endpoints

Содержит настройки прослушиваемых адресов. Детальная настройка

ServerOptions / Limits / MaxRequestBodySize

Максимальный размер тела запроса, в байтах.

Logging

Системная секция, отвечает за вывод различных логов.

GEst.Hosting / WindowsService

Наименование сервиса при его регистрации в «Службах»

GEst.Hosting / CORS

Политики CORS (Cross-origin resource sharing). Указываются источники, с которых приложение Server.Worker’а может принимать запросы.

HTTP/HTTPS WebClient – Server

В секции Endpoints находятся настройки выставленных адресов для gRPC. Url – адрес, по которому будет осуществляться общение с сервером. По умолчанию мы поставляем тестовый сертификат в

каталоге CommonFiles сервера.
Http – вариант адреса без сертификата.
Https – вариант адреса с сертификатом. Настройки сертификата указываются в секции Certificate. В секции Certificate указывается путь (Параметр Path) до pfx-сертификата и пароль от сертификата (Параметр Password).

FinistDigitalBank.Report.Server.Worker.dll.config

Настройка подключения к БД

Ожидается, что база данных уже подготовлена к использованию.

PostgreSQL

```
...
<GEst.Server>
	<ConnectionString
value="Server=pgdev.esterdev.com;Port=5432;Database=eventmanager_apptest;User
ID=postgres;Password=Nt2CFWtGaUesu;Include Error Detail=true;SSL Mode=Require;Trust
Server Certificate=true" />
	<SqlDialect value="PostgreSql" />
</GEst.Server>
...
```

server Имя или адрес сервера БД.
port Порт сервера БД.
database Наименование БД.
user id Пользователь БД, под которым будет осуществляться взаимодействие приложения с БД.
password Пароль пользователя.
SSL Mode Режим шифрования. Подробнее в документации. Обычно Require или Allow.
trust server certificate Доверять серверному сертификату. Указывает, что серверный сертификат является доверенным.
Применимо, когда на машине, где запущен севрер-приложение сертификат не добавлен в доверенные. Например, у банка есть самоподписный сертификат.

Настройка авторизации

Для работы аутентификации через доменную учётную запись, необходимо добавить в секцию GEst.Server следующую строку.

```
<GEst.Server>
	...
	<WinAuthenticationServer port="5102"/>
	...
</GEst.Server>
```

Так же, необходимо пользователю системы добавить логин вида:
domain\user + взвести флаг «Windows-аккаунт».

В случае, если серверная ОС Linux – необходимо добавить **spn**:

```
<GEst.Server>
	...
	<WinAuthenticationServer port="5102"
spn="DEMOAPP/lxnet.esterdev.com@ESTERDEV.COM"/>
	...
</GEst.Server>
```

Настройка SPN

В данном блоке описывается порядок действий, совершенный на тестовой среде одним из разработчиков, для настройки spn. В случае, если регистрация spn производится по данной инструкции, необходим скрипт **kt.cmd**. *В случае, если файла нет, его необходимо запросить.*

Прежде чем приступать к настройке, нужно убедиться что:
-DNS имя резолвится в обе стороны.
-Контроллер домена должен быть авторизованным источником времени в домене, либо все машины должна пользоваться одним источником времени.
-Linux-машину необходимо добавить в DNS, имя должно резолвиться в обе стороны.

Предоставлены следующие машины:
Контроллер домена dc1.testlab.local Windows Server 2019, домен testlab.local (NETBIOS имя TESTLAB)
Рабочая станция Windows w10.testlab.local Windows 10 21H2
Рабочая станция Linux u20.testlab.local Ubuntu 2004

На контроллере домена, с правами администратора домена выполняем следующие действия:
Создаём сервисного пользователя labsvb, указываем пароль и опции **password never expires** и **user can't change password**. В настройках пользователя указываем
This account supports Kerberos AES 128 bit encryption и **256 bit encryption**.
Скрипт, для присваивания SPN и формирования keytab-файла называется **kt.cmd**, должен быть приложен к письму. *В случае, если файла нет, его необходимо запросить.*
В начале файла изменить строчки:

```
set "KUSER=TESTLAB\labsvc"
set "KPASS=pass123"
set "KRESETPASS=no"
set "KSPN=DEMOAPP/u20.testlab.local;DEMOAPP/u20"
```

KUSER – имя сервисного пользователя.
KPASS – пароль сервисного пользователя.
KSPN – список spn для регистрации.

В данном примере мы зарегистрируем два SPN:
DEMOAPP/u20.testlab.local@TESTLAB.LOCAL
DEMOAPP/u20@TESTLAB.LOCAL
Создаём каталог на контроллере домена, например «keytabs», подкладываем туда скрипт изменённый скрипт **kt.cmd** и исполняем его.
В результате должны быть зарегистрированы spn-ы, а в каталоге сформироваться файл labsvc.keytab.
Файл .keytab подкладываем на linux-машину.

Проверить, что SPN зарегистрирован можно командой `setspn -L labsvc`.
На Linux-машине настраиваем FQDN, имя хоста - u20.testlab.local, в качестве резолвера указать контроллер домена.

Добавляем SPN в секцию **WinAuthenticationServer**

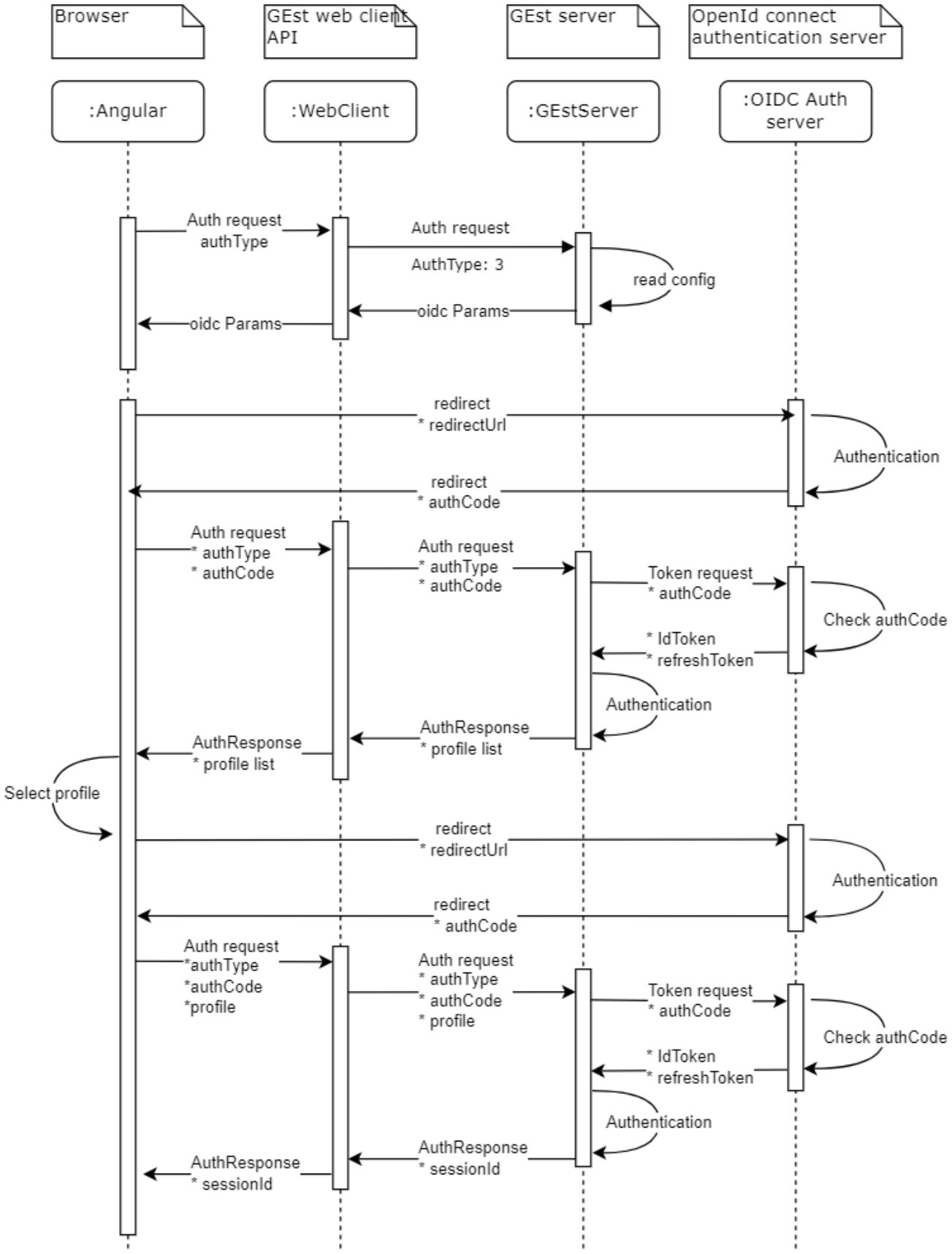
```
<GEst.Server>
```

```
      ...
      <WinAuthenticationServer port="5102"
spn="DEMOAPP/u20.testlab.local@TESTLAB.LOCAL"/>
      ...
</GEst.Server>
```

Настройка OpenId

В сервер аутентификации банка необходимо добавить нового OIDC клиента, который содержит **обязательный** scope **openid**, в результате должно быть известно два поля – **ClientId** и **ClientSecret**.
Для большинства полей, необходимо взаимодействие с банком.
Данные подключений/идентификаторов и секреты клиентов мы получаем от них.

Схема взаимодействия



ADFS

Необходимо добавить новую группу приложений в Application Groups и получить ClientId/ClientSecret.

Запускаем Server Manager
В меню Tools выбираем AD FS Management
В левой панели, в дереве ПКМ на папке **Application Groups** и нажмите на **Add Application Group…**
В следующем окне введите имя группы и, если необходимо, описание.
Выберите шаблон **Server application accessing a web API** и нажмите Next.
В открывшемся окне вы увидите **Client Identifier**, это значение нужно скопировать и поместить его в конфигурационный файл настроек как значение параметра **ClientId**. В поле Redirect URI необходимо вставить url, по которому доступен тонкий клиент с постфиксом **/atoken** и далее нажать **Add**.
Жмём кнопку **Next** и переходим к следующему шагу.
Отмечаем галочкой пункт **Generate a shared secret**. Тут нужно скопировать сгенерированный **Secret** и поместить его в конфигурационный файл настроек как значение параметра **ClientSecret**.
В следующем окне, в поле **Identifier** введите **ClientId**, который вы получили выше, нажмите добавить.
На следующей странице настройте политики доступа до клиента GEst. Это может быть определённая группа пользователей, либо все пользователи.
Далее, необходимо выбрать определённые **scopes**, доступные для клиента. В данном случае будет достаточно следующих: **openid**, **allatclaims**
Этап настройки и подготовки клиента ADFS для работы с тонким клиентом завершён. Далее, необходимо настроить политики **CORS** ADFS'a

Добавить CORS-политику, где указать источник*****, на котором располагается тонкий клиент как доверенный.
***https://domain.com** без параметров запроса.

Это делается командами внутри PowerShell'a:
включаем возможность работы с CORS
Set-AdfsResponseHeaders -EnableCORS $true
добавляем адрес, на котором расположен тонкий клиент в доверенные
Set-AdfsResponseHeaders -CORSTrustedOrigins https://domain.com

Можно приступать к конфигурации сервера.

Keycloak

Необходимо добавить **Client, его Id и будет нашем ClientId.**

Указать ему Client authentication, Standard flow и Direct access grants в On:

Сохранить, появится вкладка «Credentials». В ней указать тип **Client Authenticator** в **Client Id and Secret.**
Появится поле **Client Secret**. Копируем его.

Так же, необходимо указать источники, с которых будет осуществляться запрос к серверу аутентификации:
Добавить CORS-политику, где указать источник*****, на котором располагается тонкий клиент как доверенный. ***https://domain.com** без параметров запроса.

Можно приступать к конфигурации сервера.

Конфигурация сервера

Далее, необходимо добавить и сконфигурировать секцию **OpenIdServer** в серверном конфигурационном файле FinistDigitalBank.Report.Server.Worker.dll.config:

```
<GEst.Server>
...
        <OpenIdConnect defaultServer="server">
                <Server key="server"
                                displayName="Some server (OpenId Connect)"
                                host="oidc.server.ru"
                                useSsl="true"
                                port="443"
                                repeatedAuthenticationInterval="0:10:0">
                        <GEstAuthentication scope="openid"
                                clientId="CLIENT_ID"
            clientSecret="CLIENT_SECRET"
                                resource="urn:microsoft:userinfo"
                                responseType="code"
                                grandType="authorization_code">
                        </GEstAuthentication>
                </Server>
        </OpenIdConnect>
...
</GEst.Server>
```

*Серверов можно указать несколько. Стандартный указывается в поле **defaultServer**, где ключ = **key** сервера.*

host

Адрес который содержит каталог .well-known и конфигурационный файл oidc.

useSsl

Проверять сертификат сервера на подлинность.

port

Порт, на котором развёрнут сервер аутентификации.

Конфигурация пользователя

Каждому пользователю системы, для которого предполагается возможность аутентификации по OpenId Connect протоколу, необходимо в приложении добавить новый логин с наименованием, которое будет в точности соответствовать имени пользователя (без указания доменной части), под которым происходит аутентификация на сервере аутентификации и включенным крыжиком «Windows-аккаунт».

LDAP

В серверном конфигурационном файле (FinistDigitalBank.Report.Server.Worker.dll.config) есть возможность настроить LDAP сервер, для аутентификации пользователей через доменные учётные данные.

LDAP серверов может быть несколько.

```
<Ldap>
    <!--
                key - уникальный ключ сервера; обязательный параметр; сервер с ключом some_server будет
обрабатывать аутентификацию пользователей с
                    именами вида some_server\имя_пользователя; обязательный параметр
                host - адрес сервера LDAP; обязательный параметр
                port - порт для подключения; обязательный параметр
                useSsl - флаг использования SSL; по умолчанию false
```

```
                trustServerCertificate - флаг отключения проверки серверного сертификата; работает только при
useSsl="true"; по умолчанию false
                repeatedAuthenticationInterval - интервал повторной аутентификации пользователей; формат -
System.TimeStamp; не может превышать 1 день; по умолчанию 10 минут
                authenticationType - тип аутентификации; значение соответствует enum'у
System.DirectoryServices.Protocols.AuthType; по умолчаннию Basic
                protocolVersion - версия протокола LDAP; по умолчанию 3
                displayName - отображаемое имя сервера
            -->
    <Server key="some_server" host="some_server.com" port="389" useSsl="false" trustServerCertificate="false"
repeatedAuthenticationInterval="0:10:0"
                authenticationType="Basic" protocolVersion="3" displayName="Some server">
        <GEstAuthentication>
            <!-- Настройки операции LDAP bind (первоначальная аутентификация, используется для поиска
аутентифицированного пользователя) -->
            <!-- dynamic - флаг использования учетной записи текущего пользователя для первоначальной
аутентификации; обязательный параметр
                 dn - distinguished name учетной записи пользователя, использующейся для первоначальной
аутентификации; если dynamic="true", то может содержать плейсхолдер
                    логина текущего пользователя %%login%%; для пользователя some_server\some_user %%login%%
== some_user; обязательный параметр
                 password - пароль учетной записи пользователя, использующейся для первоначальной
аутентификации; используется только если dynamic="false"; необязательный параметр -->
            <Bind dynamic="true" dn="uid=%%login%%,ou=Users,dc=some_server,dc=com" password="some_password" />
            <!-- Настройки LDAP query, использующегося для поиска аутентифицированного пользователя -->
            <!-- root - distinguished name каталога, с которого начинается поиск; обязательный параметр
                 filter - LDAP запрос для поиска пользователя; д.б. составлен так чтобы не вернуть больше
одной записи (нужен uid/sAMAccountName); не должен
                 возвращать заблокированных пользователей (критерии нужно согласовать с администратором
сервера); обязательный параметр -->
            <Search root="dc=some_server,dc=com" filter="(&amp;(uid=%%login%%)(objectClass=posixAccount))" />
        </GEstAuthentication>
    </Server>
    <!-- Пример настройки сервера для аутентификации в ActiveDirectory (на базе AD esterdev) -->
    <Server key="esterdev" host="bighead.esterdev.com" port="389" useSsl="false"
trustServerCertificate="false" repeatedAuthenticationInterval="0:10:0" authenticationType="Basic"
protocolVersion="3" displayName="EsterDev (LDAP)">
        <GEstAuthentication>
            <Bind dynamic="true" dn="%%login%%@esterdev.com" />
            <Search root="dc=esterdev,dc=com"
filter="(&amp;(sAMAccountName=%%login%%)(objectCategory=person)(objectClass=user)(!(userAccountControl:1.2.840
.113556.1.4.803:=2)))" />
        </GEstAuthentication>
    </Server>
    <!-- Пример настройки сервера для аутентификации в openLDAP (на базе локальной тестовой платформы) -->
    <Server key="ldaptest" host="192.168.1.114" port="389" useSsl="false" trustServerCertificate="false"
repeatedAuthenticationInterval="0:10:0" authenticationType="Basic" protocolVersion="3">
        <GEstAuthentication>
            <Bind dynamic="true" dn="uid=%%login%%,ou=Users,dc=ldaptest,dc=com" />
            <Search root="dc=ldaptest,dc=com" filter="(&amp;(uid=%%login%%)(objectClass=posixAccount))" />
        </GEstAuthentication>
    </Server>
</Ldap>
```

Для возможности входа без указания доменного имени, необходимо указать атрибут **defaultServer** - уникальный ключ сервера, значение из секции **Server/@key**. Этот домен будет использоваться как основной. Пользователи, **расположенные в нём**, могут не указывать доменное имя при авторизации.

```
<GEst.Server>
        ...
        <Ldap defaultServer="orgdomain" ...>
                <Server key="orgdomain">
                ...
                </Server>
                <Server key="otherdomain" ...>
                ...
                </Server>
```

```
        </Ldap>
        ...
</GEst.Server>
```

FinistDigitalBank.Report.WebClient

Конфигурируется двумя файлами.
FinistDigitalBank.Report.WebClient.dll.config – основной файл конфигурации. Используется для настройки общения веб-приложения с сервером.
Appsettings.json – файл конфигурации для настройки общения веб-приложения с тонким клиентом.

FinistDigitalBank.Report.WebClient.dll.config

```
...
<GEst.Client>
    ...
    <Credential type="Basic" login="razrab" password="papapa" profile="Базовый
профиль"/>
</GEst.Client>
<GEst.Common>
    <HostAddress value="http://localhost:5700" />
</GEst.Common>
...
```

В данном файле нас интересует секция **GEst.Client** и **GEst.Common**.
HostAddress
Адрес сервера. Настраивается в **appsettings.json** сервера. По нему будет осуществляться общение веб-приложение – сервер.
Credential
Настройки авторизации веб-приложения на сервере.
Есть два типа авторизации:
Basic – авторизация по связке логин-пароль пользователя **внутри системы опер.рисков**.

- Login – Логин пользователя.
- Password – Пароль пользователя.
- Profile – Наименование профиля пользователя в системе.

Пользователь должен обладать минимальным уровнем доступа для корректной работы.
SSPI – Авторизация через windows-аккаунт. Настройка производится на сервере.

Для первого запуска используется сервисная учётная запись razrab/papapa, в последствии, её можно заменить на другую.

Appsettings.json

```
{
    "AllowedHosts": "*",
    "Kestrel": {
        "Endpoints": {
            "Http": {
                "Url": "http://0.0.0.0:5050"
            }
            //"Https": {
            // "Url": "https://0.0.0.0:5051"
            // Uncomment following section for use specified certificate for API hosting.
            //"Certificate": {
            //    "Path": "CommonFiles/DevRiskCert.pfx",
            //    "Password": "DevRisk"
            //}
            // }
        },
        "ServerOptions": { // GEst.Hosting options
```

```
            "Limits": {
                "MaxRequestBodySize": 157286400
            }
        }
    },
    "GEst.Hosting": {
        //"WindowsService": { // only for Windows services
        //    "ServiceName": "MyWinService" // by default application name (name of entry assembly
or startup assembly(for web))
        //},
        "CORS": { // CORS policies - will be added if specified at least one with nonempty name
            "Policies": [{
                    "Name": "GEst.WebClient.Api", // for api controllers in GEst.WebClient
                    "Origins": ["http://localhost:4205", "http://localhost"]
                }
            ]
        }
    }
}
```

Endpoints
Настройки выставленных адресов для gRPC. Url – адрес, по которому будут приниматься запросы от тонкого клиента.
Http
Настройка прослушиваемого адреса без сертификата.
Https
Настройка прослушиваемого адреса с сертификатом.
Настройки сертификата указываются в секции **Certificate**. В секции указывается путь **Path** до pfx-сертификата и пароль от сертификата **Password**.
ServerOptions / Limits / MaxRequestBodySize
Максимальный размер тела запроса, в байтах.
GEst.Hosting / CORS
Политики CORS (Cross-origin resource sharing). Указываются источники, с которых приложение WebClient.Worker'а может принимать запросы.

Дополнительная информация о WebClient

Для изменения прав сервисной учётной записи - необходимо из толстого клиента (UserArm) настроить учётные записи, под которыми будет осуществляться работа веб-приложения.
Учётная запись должна обладать правами доступа уровня «Административный»
После создания пользователя – перезапустить систему безопасности (Сервис – Перезапуск системы безопасности).

FinistDigitalBank.Report.WDS

Набор файлов тонкого клиента.
В этом наборе можно заменить файлы:
favicon.ico – значок веб-сайта в браузере.
Так же, в каталоге assets можно заменить/редактировать следующие файлы:
logo.png – логотип приложения на странице входа.
styles/global-colors.css – цвета приложения.

Файлы из архива необходимо распаковать в каталог **public_html** веб-сервера.

Обновление файлов WDS

1. Необходимо удалить старые файлы **кроме** каталога assets.
2. Распаковать архив с тонким клиентом (без папки assets).

Конфигурация тонкого клиента

С версии тонкого клиента 1.4.24+ появился файл конфигурации.

В каталоге **assets/configuration** содержится базовая конфигурация **default.json:**

```
{
	"DefaultTitle": "СУОР",
	"CombineTitle": false,
	"ShowBasicAuth": true
}
```

DefaultTitle – стандартное наименование вкладки, если не выбрано какое-либо окно

CombinedTitle – флаг, указывающий, что **DefaultTitle** и текущую страницу необходимо комбинировать

ShowBasicAuth – показывать стандартную страницу ввода логина/пароля:

Данный флаг применим, когда предусмотрена авторизация через [протокол OpenID.]

Настройка веб-сервера

Где находится веб-сервер?

Linux

В линуксе, с помощью команды whereis nginx | grep etc находим путь, содержащий /etc/

```
[root@fin-hl-app ~]# whereis nginx | grep etc
nginx: /usr/sbin/nginx /usr/lib64/nginx /etc/nginx /usr/share/nginx /usr/share/man/man3/nginx.3pm.gz /usr/share/man/man8/nginx.8.gz
[root@fin-hl-app ~]#
```

В этом каталоге и будет находиться файл конфигурации nginx’a.

Стандартный файл конфигурации nginx находится в каталоге **conf**. Нас интересует секция http / server:

```
server
{
    listen 80;
    server_name _;

    # путь до файлов тонкого клиента
    root home/localhost/public_html;
    index index.html;

    log_not_found off;
    # логи, в которые пишется история доступов к веб-серверу
    access_log logs/web-access.log;


    # все остальные запросы возвращают 401.
    location ~ /\. { deny all; }
    location = /favicon.ico { }
    location = /robots.txt { }
}
```

Далее, для работы тонкого клиента, нам необходимо перенаправлять (проксировать) все запросы к веб-серверу по маске “/api/”, к нашему приложению WebClient’a:

```
# маска для перенаправления запроса
location /api/
{
   proxy_pass https://адрес-webclient:webclient-порт$request_uri;
   proxy_redirect off;
}
```

!Важно! Если в WebClient настроена только секция http, то протокол взаимодействия должен быть http. Иначе – https.

В результате должно получиться:

```
server
{
    listen 80;
    server_name _;

    #Путь до файлов тонкого клиента
    root home/localhost/public_html;
    index index.html;

    log_not_found off;
    # логи, в которые пишется история доступов к веб-серверу
    access_log logs/web-access.log;

    # маска для перенаправления запроса
    location /api/
    {
        proxy_pass http://адрес-webclient:webclient-порт$request_uri;
        proxy_redirect off;
    }



    # все остальные запросы возвращают 401.
    location ~ /\. { deny all; }
    location = /favicon.ico { }
    location = /robots.txt { }
}
```

Конфигурация для работы с OpenId/ADFS

Для работы с OpenId/ADFS необходимо добавить дополнительное перенаправление:

```
server
{
    listen 80;
    server_name _;

    #Путь до файлов тонкого клиента
    root home/localhost/public_html;
    index index.html;

    log_not_found off;
    # логи, в которые пишется история доступов к веб-серверу
    access_log logs/web-access.log;

    # маска для перенаправления запроса
    location /api/
    {
        proxy_pass http://адрес-webclient:webclient-порт$request_uri;
        proxy_redirect off;
    }

    # маска для перенаправления запроса из OpenId/ADFS
    location /atoken
    {
        return 301 $scheme://$host/#$request_uri;
    }



    # все остальные запросы возвращают 401.
    location ~ /\. { deny all; }
    location = /favicon.ico { }
    location = /robots.txt { }
}
```

Конфигурация HTTPS

Для подписания веб-сайта сертификатом потребуется связка публичный сертификат и его приватный ключ. Сертификаты, в общем случае, мы получаем от сотрудников банка.
Необходимо переконфигурировать веб-сервер

Изменить прослушиваемый порт с 80 на 443 и добавить аргумент **ssl**.

```
    listen 443 ssl;
```

Добавить свойства для указания сертификатов:

```
    # сертификаты для HTTPS
    ssl_certificate     www.example.com.crt;
    ssl_certificate_key www.example.com.key;
```

Добавить поддерживаемые методы шифрования:

```
    ssl_protocols       TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;
    ssl_ciphers         HIGH:!aNULL:!MD5;
```

В результате должно получиться:

```
server
{
    listen 443 ssl;
    server_name _;

    #Путь до файлов тонкого клиента
    root home/localhost/public_html;
    index index.html;
    # Максимальный вес запроса (заголовки + вес тела), увеличить для загрузки больших файлов.
    client_max_body_size 50m;


    log_not_found off;
    # логи, в которые пишется история доступов к веб-серверу
    access_log logs/web-access.log;

    # сертификаты для HTTPS
    ssl_certificate     www.example.com.crt;
    ssl_certificate_key www.example.com.key;
    ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;

    # маска для перенаправления запроса
    location /api/
    {
        proxy_pass http://адрес-webclient:webclient-порт$request_uri;
        proxy_redirect off;
    }

    # маска для перенаправления запроса из OpenId/ADFS
    location /atoken
    {
        return 301 $scheme://$host/#$request_uri;
    }



    # все остальные запросы возвращают 401.
    location ~ /\. { deny all; }
    location = /favicon.ico { }
    location = /robots.txt { }
}
```

Перенаправление HTTP => HTTPS

В предыдущем пункте мы настроили веб-сервер для работы по протоколу https, это привело к тому, что при открытии по прямой ссылке «http://СУОР» у нас будет ошибка. Что бы этого избежать – необходимо добавить перенаправление с http на https.

Для этого необходимо отредактировать файл конфигурации nginx.conf:

В блоке http необходимо добавить ещё одну секцию server, помимо той, что мы настроили для https:

```
server
{
    listen 80;

    return 301 https://$host$request_uri;
}
```

Конечный файл, с настройками HTTP и HTTPS будет выглядеть так.

Пример файла конфигурации

```
server
{
    listen 80;
    # редирект на хост с протоколом https
```

```
    return 301 https://$host$request_uri;
}

server
{
    listen 443 ssl;
    server_name _;

    # путь до файлов тонкого клиента
    root home/localhost/public_html;
    index index.html;
    # Максимальный вес запроса (заголовки + вес тела), увеличить для загрузки больших файлов.
    client_max_body_size 50m;


    log_not_found off;
    # логи, в которые пишется история доступов к веб-серверу
    access_log logs/web-access.log;

    # сертификаты для HTTPS
    ssl_certificate     www.example.com.crt;
    ssl_certificate_key www.example.com.key;
    ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;

    # маска для перенаправления запроса
    location /api/
    {
        proxy_pass http://адрес-webclient:webclient-порт$request_uri;
        proxy_redirect off;
    }

    # маска для перенаправления запроса из OpenId/ADFS
    location /atoken
    {
        return 301 $scheme://$host/#$request_uri;
    }



    # все остальные запросы возвращают 401.
    location ~ /\. { deny all; }
    location = /favicon.ico { }
    location = /robots.txt { }
}
```

Проверка конфигурации Nginx

Прежде чем перезапускать сервис, проверим, что заполнили конфигурацию корректно. Это можно сделать командой `nginx -t`

Перезапуск и применение изменений Nginx

Linux

Изменения применяются после перезапуска сервиса. Сделать это можно командой

`systemctl restart nginx`

В случае успешного перезапуска – в консоль ничего не выведется.

Иначе, будет выведена информация о том, что при запуске сервиса возникли ошибки

```
[root@fin-hl-app ~]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@fin-hl-app ~]# systemctl restart nginx
Job for nginx.service failed because the control process exited with error code. See "systemctl status nginx.service" and "journalctl -xe" for details.
[root@fin-hl-app ~]#
```

Регистрация сервисов

Теперь в службах зарегистрирован nginx.exe. Он будет запускаться вместе с системой.

Linux

Регистрация приложений

В ОС Linux сервисы (юниты) регистрируются путём создания файла-юнита.
Все операции должны быть выполнены root-пользователем (команда sudo).
Пример юнита для EventManager.WebClient.Worker.dll

```
[Unit]
Description=EventManager Web Client Service

[Service]
Type=notify
# will set the Current Working Directory (CWD). Worker service will have issues without
this setting
WorkingDirectory=PATH_TO_APP_DIRECTORY

# systemd will run this executable to start the service
ExecStart=/usr/bin/dotnet PATH_TO_WORKER_DLL &

# to query logs using journalctl, set a logical name here
SyslogIdentifier=EventManager.WebClient.Worker

# Use your username to keep things simple.
# If you pick a different user, make sure dotnet and all permissions are set correctly
to run the app
# To update permissions, use 'chown yourusername -R
/opt/EM/EventManager.WebClient.Worker' to take ownership of the folder and files,
#       Use 'chmod +x
/opt/EM/EventManager.Server.Worker/EventManager.WebClient.Worker.dll' to allow execution
of the executable file
User=adm

# ensure the service restarts after crashing
# Restart=always
# amount of time to wait before restarting the service
# RestartSec=15

# This environment variable is necessary when dotnet isn't loaded for the specified
user.
# To figure out this value, run 'env | grep DOTNET_ROOT' when dotnet has been loaded
into your shell.
# Environment=DOTNET_ROOT=/opt/rh/rh-dotnet31/root/usr/lib64/dotnet
# Environment=DOTNET_ROOT=/usr/bin/dotnet

[Install]
WantedBy=multi-user.target
```

Основные конфигурационные поля:

WorkingDirectory - каталог содержащий файлы и конфиги сервиса.

ExecStart – Исполняемая команда при запуске сервиса через systemctl. Указываем путь до исполняемой dll.

Конфигурационный файл применим как к сервису сервера, так и к сервису веб-клиента, необходимо лишь заменить пути до запускаемых файлов.

User – Пользователь, под которым будет осуществлён запуск и исполнение приложения.

***Поля отмеченные красным – необходимо заполнить в соответствии с вашей конфигурацией.**

Для создания файла нам необходимо выполнить консольную команду
`touch /etc/systemd/system/EventManager.Server.Worker.service`
Данная команда создаст пустой файл EventManager.Server.Worker.service. Далее, при помощи команды
`nano /etc/system/system/EventManager.Server.Worker.service`
откроем данный файл в редакторе nano и вставим (ПКМ) конфигурацию юнита из примера выше
Не забыв заменить пути до приложения **PATH_TO_APP_DIRECTORY** и **PATH_TO_WORKER_DLL**

После проделанных действий вызываем команду:
`systemctl daemon-reload`
которая обновит и подтянет конфигурации system.
Далее, мы можем произвести запуск через сервиса командой:
`systemctl start EventManager.Server.Worker.service`
Для проверки статуса сервиса необходимо использовать команду
`systemctl status EventManager.Server.Worker.service`
В случае, если запуск сервиса произведён успешно, будет выведено такое сообщение:
Важно проверить статус сервиса спустя несколько минут, так как для запуска приложения требуется время.

Настройка общего доступа до каталога с Linux на Windows

Перед конфигурацией linux-системы, создадим нового пользователя windows с паролем и дадим ему право на чтение каталога. Это нужно для последующего использования данной учётной записи в linux.

Для того, что бы пользоваться общим windows-каталогом на linux сервере, необходимо установить утилиту **cifs-utils.** Она позволит подключить (mount) общий каталог как отдельный диск. Под супер пользователем исполняем команду: `apt install cifs-utils`
логи во время установки пакета

Далее, создадим каталог, который будет являться точкой подключения, командой mkdir /mnt/win-share.

Тестовое подключение каталога

После создания каталога, необходимо «примонтировать» общий каталог, это делается командой
`mount -t cifs -o username=%win пользователь% //%ip windows машины%/%каталог% /mnt/win_share`

Конфигурация авто-монтирования

Для того, что бы после перезапуска системы каталог оставался доступен – необходимо указать системе, что его нужно подключать в момент запуска системы. Делается это через изменение конфигурации файловой системы, файл `/etc/fstab`.

Прежде чем приступать к изменению fstab – создадим файл, который будет содержать данные об авторизации в windows. Создадим каталог и файл с данными, командами:
`mkdir /etc/win-creds/`
`touch /etc/win-creds/share`
и откроем его при помощи **nano** (или любого другого редактора)
Указываем логин/пароль/домен пользователя (если существует).

После изменения файла, настроем доступы до каталога. Запретим всем пользователям, кроме root как-либо взаимодействовать с файлами win-creds. Это делается командами:
`chown root: /etc/win-creds`
`chmod 600 /etc/win-creds`

Теперь перейдём к редактированию fstab файла. Добавляем строчку
`//%ip win-машины%/%общедоступный каталог% /mnt/win-share cifs credentials=/etc/win-creds/share,file_mode=0755,dir_mode=0755 0 0`
Данная запись будет использована при запуске системы, а сейчас необходимо примонтировать вручную, командой `mount /mnt/win-share`
Готово. Теперь данный каталог будет монтироваться при каждом запуске системы. Так же, для удобства использования в веб-сервере, можно создать символическую ссылку, дабы не ссылаться напрямую в /mnt/

Создание символьных ссылок в Linux (симлинк)

Данный механизм может быть полезен как для поставки обновлений банку, так и для внутреннего использования. Например, для настройки веб-сервера и его root каталога.

Для создание симлинка нужно выполнить команду:
`ln -s %каталог, к которому создаётся симлинк% %каталог, куда симлинк будет помещён%`

Пример использования сим-линка:
В /home/ каталоге лежат файлы, которые пользователь может редактировать, однако, у него нет доступа до каталога /var/www/html и он не может копировать туда файлы.
Для того, чтобы пользователю не требовалось конфигурировать веб-сервер и изменять системные каталоги, мы создадим симлинк до каталога, куда у пользователя доступ имеется и при последующих изменениях ему не потребуется административный доступ:
В настройках веб-сервера же будет указана следующая строчка
Теперь веб-сервер ссылается на сим-линк, который, в свою очередь, указывает на пользовательский каталог.

Проверка работоспособности

Проверить «возникали ли ошибки при старте сервисов» можно в утилите Event Viewer (менеджер событий). Если ошибок не возникло – можно воспользоваться приложением.
Открыть веб-браузер, ввести адрес машины/глобальный адрес машины, на которой развёрнут веб-сервер.

При корректной настройке веб-сервера/указания адреса – откроется стандартная форма логина. Вход в приложение можно осуществить при помощи связки логин (с доменом) + пароль, от windows-аккаунта, в случае, если настроен LDAP сервер.

*цветовая схема и логотип могут быть изменены.

В форме указываем логин/пароль пользователя. Если всё настроено корректно, то откроется окно выбора рабочего профиля (если профилей несколько)

После выбора – нас перенесёт на основную страницу приложения:

*Выход осуществляется по нажатию иконки «пользователь», в контекстном меню будет кнопка «Выйти»

Проблемы, которые могут возникнуть

При авторизации по корректному логину/паролю ничего не происходит.

Если не возникло никаких дополнительных информационных сообщений, то, скорее всего, дело в том, что был добавлен новый логин для пользователя/пользователь. В таком случае, система безопасности не перезагружена и сервер-приложение не знает о новом логине пользователя.

При авторизации возникает ошибка 502 Bad gateway

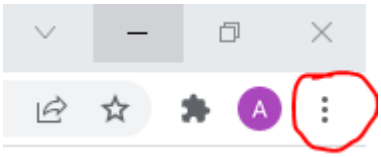
Одно из возможных решений - проверить настройку веб-сервера. Корректно ли он перенаправляет запросы.
Если запросы перенаправляются корректно, то, возможно, ошибка в настройке веб-приложения. Необходимо убедиться, что в конфигурационном файле Appsettings.json закомментирована секция https (при отсутствии сертификата).
Если секция закомментирована, но ошибка сохраняется – необходимо проверить менеджер событий Windows. Возможно, сервис запустился некорректно/возникли ошибки при запуске сервиса.

При авторизации возникает ошибка 408 Request timeout

Необходимо убедиться, что веб-приложение корректно натравлено на сервер-приложение.
Так же, проверить, что сервер-приложение запущено и работает.

При авторизации возникает ошибка 500 | Другие ошибки

Необходимо связаться с разработчиками, приложив файл .har из консоли разработчика, секции networking и лог консоли из секции console.
Сформировать эти файлы можно таким образом:
Открываем консоль разработчика (Google Chrome) меню браузера (три точки)



Секция меню «Больше инструментов», кнопка «Консоль разработчика» (Инструменты разработчика). Откроется окно с инструментами:

Во вкладке «Console» (Консоль) нажимаем ПКМ – «Сохранить как». Сформируется текстовый файл, его прикладываем к письму.
Далее, переходим во вкладку «Network» (Сеть):

Нажимаем ПКМ в выделенной области, выбираем пункт меню «Сохранить всё как HAR» (Save all as HAR with content). Сформированный файл прикладываем к письму.