

LAN Backup software design document (work in progress)

1. Document control

1.1 Version history

Date	Version	Notes	Author
08.01.2017	0.1	Specification, first draft	Claudiu Farcas
14.01.2017	0.1.1	Additional info	Claudiu Farcas
01.02.2017	0.1.2	Publishing to GitHub	Claudiu Farcas

2. Contents

1. Document control	2
1.1 Version history	2
2. Contents	3
3. Overview	4
4. Functional specifications	4
5. System components and diagram	4
5.1 System overview	4
5.2 Software components description	6
1. Web service - REST API	6
2. Web Application	6
3. Backup Agent	7
6. Used technologies	8
6.1 Web API technologies	8
6.1.1 REST API	8
6.1.2 SignalR	8
6.2 Web Application technologies	8
6.2.1 Server side technologies	8
6.2.2 Client side technologies	8
6.3 Backup Agent side technologies	9
7. Data models and Database schema	9
7.1 Data Model	9
BackupConfiguration entity model	9
BackupLog entity model	10
7.2 Database schema	11
8. User Interface – web application	11
8.1 Login view	11
8.2 Register view	12
8.3 Dashboard (home) view	12
8.4 Backups view	12
8.5 Users management view	14
9. Software project structure	15
10. System security	15
11. Deployment and Configuration	16

3. Overview

The goal of the system is to allow centralized backup of certain specified data (files and folders) into designated storage locations, inside internal network of the organization.

4. Functional specifications

The system allows centralized management and monitoring of backups in a corporate LAN. The requirement is creation of a system where backups can be scheduled for LAN machines. The machines are windows based.

To include a machine into the backup infrastructure, an agent/client must be installed that does actual backups to some shared network folder.

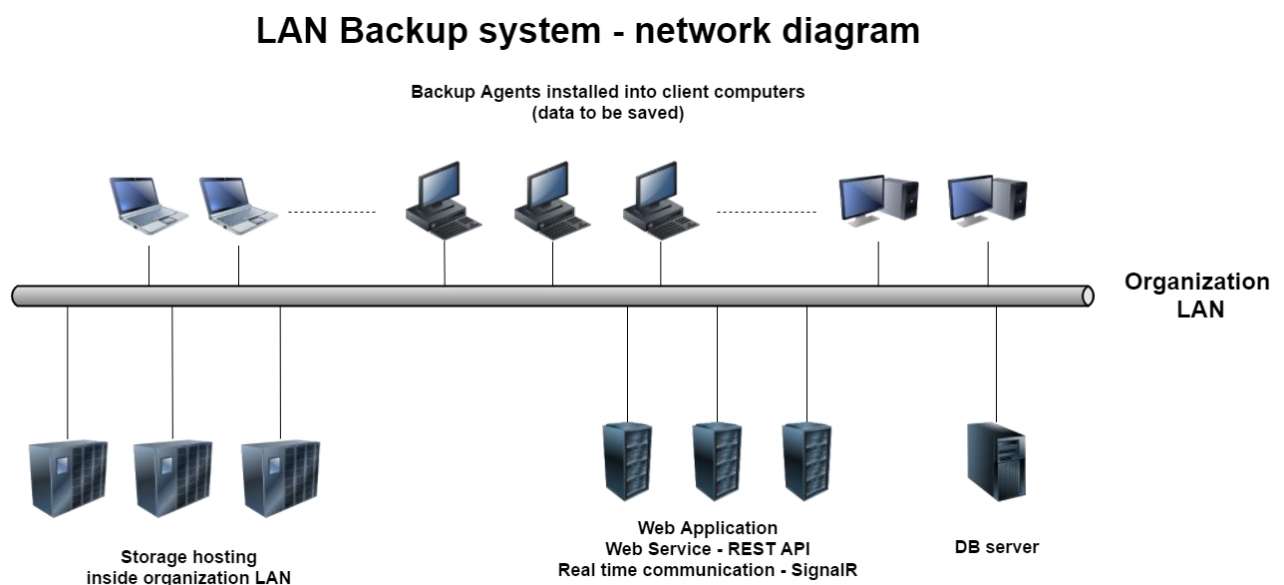
5. System components and diagram

5.1 System overview

The system is composed of the following major functional components:

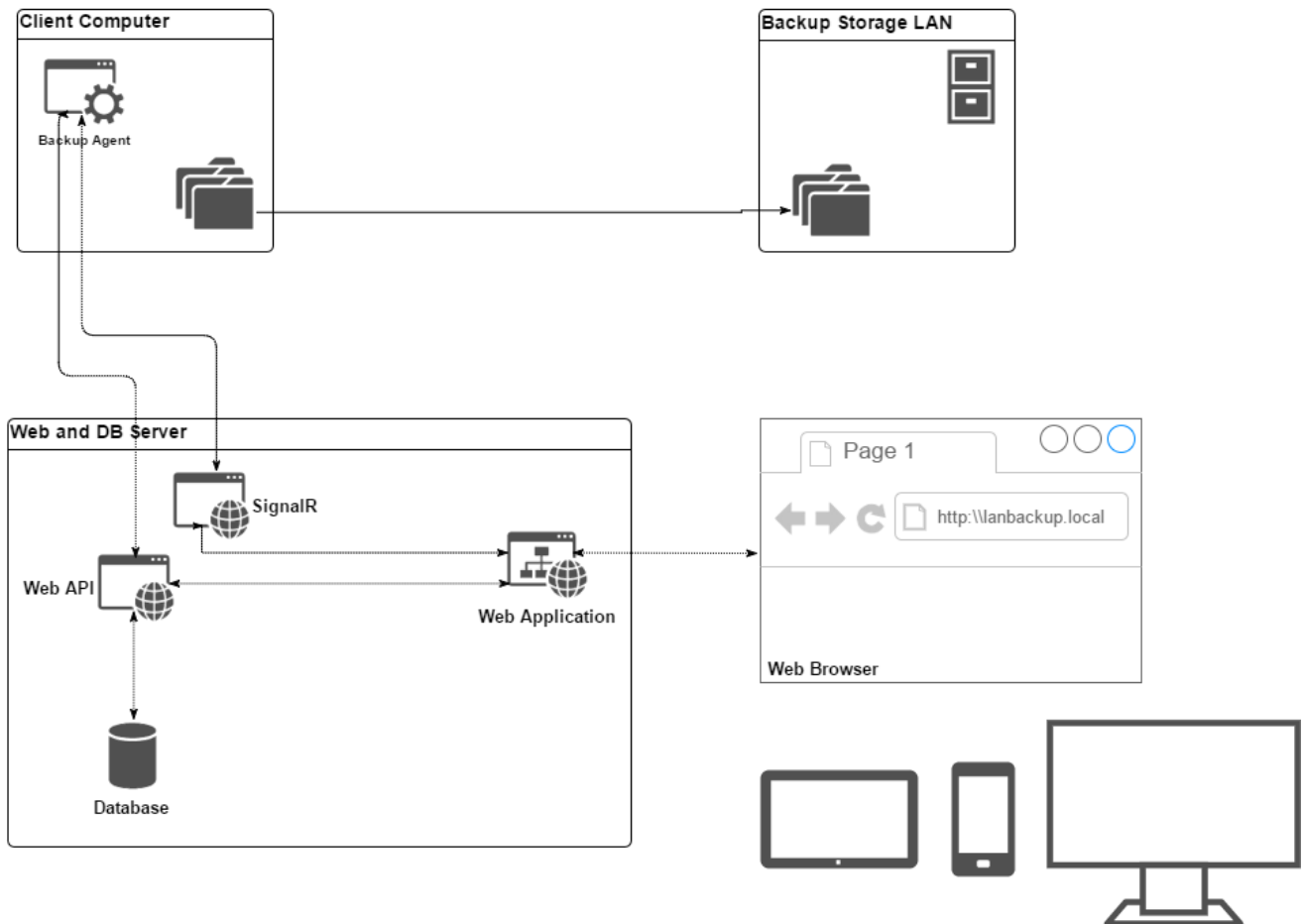
1. A web server that will host the web application accessible inside corporate LAN, this server will also host REST services and other real-time notification services needed to support functionality.
2. A database storage machine; it can be the same machine as the web server if the operations volume is low. It is recommended to migrate to a dedicate server if the transactions volume increase significantly that can affect the normal performance of the web server.
3. Backup agents that will be installed into all computers where information to be saved resides. These agent's operations will be orchestrated and monitored by the central service (web app and web api).

Physically these functional components are distributed as depicted in the following diagram:



Logically these functional components and their subcomponent will interact as shown in the next diagram:

LAN Backup System - Components Interactions



5.2 Software components description

The system will have following software components:

1. **Web service - REST API** – with following roles:

- Handle CRUD operations for backup configurations - Allow configuration and management of backups on LAN machines (create, edit, enable, disable, delete).
- **Backup configuration** have the following information:
 - Client machine ip-address
 - Source folder path on the client
 - Destination folder path on a network share
 - User/password pairs to access source folders. (local computer)
 - User/password pairs to access destination folders. (LAN computer)
 - State of the configuration: Enabled/Disabled
 - Schedule information (CRONTAB format)
- Should have an endpoint where to request information about the backups for specific client machines, by ip-address (and not by any storage primary key).
- Store Agents activity into a **LOG** that can be:
 - Updated with new records from agents.
 - Queried(view) to view and analyze specific agent activity.
- All these backup configurations operations should be logged into the **LOG**.
- (Optional) Offers current state of the Agent with a SignalR endpoint (e.g.: it is idle, or it is executing a specific backup job; eventually having a progress counter)
- (Optional) Notifies backup agents about changes performed on their specific records with a SignalR endpoint (like updating an existing config, or a new config has been created etc.).
- Should support role based identity authorization with following Roles:
 - Admin – can:
 - Create, Update, Enable/Disable, Delete backup configurations
 - Promote/demote existing registered users to “Admin” role
 - User (and anonymous) – can only view current list of backups, logs and the state of each agent.
- For a user to get granted an “Admin” role it must first register and it will receive automatically “User” role. Later, only another “Admin” role user can promote it to have extra grants.

2. **Web Application** – with roles (all backend operations should be done via REST API):

- Presents following web pages (areas):
 - **Login/Register view** users:
 - Login
 - a. display only when no user is already logged.
 - b. should perform validation via REST API and if:
 - i. success – continues towards protected area of the website
 - ii. failure – allow user to retry and eventually retrieve lost/forgot password
 - Register
 - a. Should ask relevant info to be able to create an account via REST API
 - b. Once account is created successfully automatically authenticate user and navigate toward protected resource he is trying to access.
 - **Account management view** – operations performed for self-account like:
 - Changing password.
 - **Users roles management view**
 - Only “Admin” users can see and perform actions in this view.
 - An Admin user can promote or demote another user.
 - Current logged user cannot perform promote/demote against himself.

- **Backup configurations** view and management:
 - Any user can see the list of backup configurations.
 - Only users in “Admin” role can create, Update (e.g. Enable/Disable), Delete a backup configuration.
- **Logs view**
 - All users (registered and anonymous) can see the logs list.
- (Optional) **Dashboard view** where it can present current state of available agents (idle, working: progress status)

3. Backup Agent (client) application – with following roles:

- It will be installed as a service on client computers, it will be automatically started when computer starts.
- It will work silently in the background and it will not interact with current logged user.
- It will query the REST API regularly to retrieve the list of scheduled backup configurations filtered by current machine IP.
- It will evaluate periodically these schedules and when each are due it should trigger the backup operation.
- Only one backup operation will be executed once and will be performed into another thread.
- If multiple schedules conflicts or the due schedule triggers while another backup is pending these will be queued and be executed ASAP previous finishes.
- The execution of the backup operation will be logged locally (into a log file or event log) and reported into the REST API – LOG endpoint with following actions:
 - Backup due – when the schedule triggers;
 - Backup started – when it starts the backup operation, just before performing any copy over the network.
 - Backup end – when backup ended. It should report the end state:
 - Success – when all went fine.
 - Failure – documented with error message in case something went wrong.
- (Optional) Report current state to WEB API - SignalR endpoint:
 - Idle – not performing any operation
 - Working: specifying for which schedule and progress status
- (Optional) Keep contact with WEB API service via SignalR endpoint and get notified when some client regarding data has changed. When this happens, the agent should request fresh info regarding its own scheduled configurations and act appropriately:
 - For schedule configurations that are disabled or have been deleted (not present in the response of REST API) – if there is a backup ongoing it should be stopped (and reported).
 - Empty current working queue and reevaluate fresh retrieved schedules. This will re-populate the queue if schedules are due.

6. Used technologies

We should aim in using those technologies that will offer the most efficient design.

6.1 Web API technologies

For web API we have opted for ASP.NET WEB.API CORE as this technology has numerous advantages over the previous versions of ASP.NET (quick comparison here <https://www.asp.net/learn>)
<https://www.asp.net/web-api>

6.1.1 REST API

The REST API will be developed in WEB.API CORE technologies presented earlier.
Having REST API available as a separate endpoint make it very convenient for other consumers (like the web applications, backup agents or even other applications) to consume exposed services.

6.1.2 SignalR

Even if SignalR is considered a legacy of previous ASP.NET stack, the new ported CORE version works fine with CORE projects and leverages the power of real time communication of SignalR between the distributed components.

By implementing SignalR on server side as well on the client side we benefit of real time notification:

- Agents get notified when backup configurations have changed, added, or deleted.
- Web user interface gets updated in real time with the status of each backup agent.

6.2 Web Application technologies

On Web Application stack, we have opted for various technologies that can be separated in two categories based on where they are running: server side and client side.

6.2.1 Server side technologies

On server side, we have opted for latest ASP.NET CORE MVC.
ASP.NET Core MVC is a model view controller framework for building dynamic web sites with clean separation of concerns, including the merged MVC, Web API, and Web Pages w/ Razor
Project site : <https://github.com/aspnet/Mvc>

6.2.2 Client side technologies

Client technologies stack:

- HTML5
- JavaScript frameworks:
 - AngularJS - On the client side, we have opted for AngularJS version 2 (project site: <https://angular.io/>).
AngularJS is a structural framework for dynamic web apps. It will help in creating this web application as SPA application.

Single-Page Applications (SPAs) are Web apps that load a single HTML page and dynamically update that page as the user interacts with the app.

In our application, the client side will call REST API to handle most of the operations.

- CSS and JS frameworks:
 - **Bootstrap** - a free and open-source front-end web framework for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. (project site: <http://getbootstrap.com/>)
 - **jQuery** - as part of bootstrap package; it is a fast and concise JavaScript Library created with a nice motto 'Write less, do more'. (project site: <https://jquery.com/>)
- Helpers:
 - **Webpack** - is a module bundler used to pack and minify production use of CSS and JS (project site: <https://webpack.js.org/>)

6.3 Backup Agent side technologies

Backup agent is a windows service created in standard .NET.

It makes use of http to access REST API to retrieve client specific data or to update the log.

Also it uses standard .NET SignalR libraries to communicate with server side SignalR to get notifications or to notify the server (and others) about its current working state.

We use of **Hangfire** library for scheduling recurrent backups (<http://hangfire.io/>).

7. Data models and Database schema

7.1 Data Model

Based on the requirements of this project we need to persist some data and info that can be grouped in following entities:

BackupConfiguration entity model

Field	Type	Description
ID	(string) Identity Primary Key	Should be automatically generated and maintained by database store. Should be unique per collection.
ClientIP	(string) Index	Represents the agent (computer) IP address. It is needed when an agent is querying for its own backup configurations. Should be unique per collection.
SrcPath	(string)	Source folder path on the client computer from where agent will copy contained files and subfolders. Should be a valid local path (e.g. C:\Projects\MainDataProject)
SrcUser	(string)	User name credential needed when impersonating for gaining access to local files and folders on client computer.
SrcPass	(string)	User password credential needed when impersonating for gaining access to local files and folders on client computer. (optional) - It is advised not to be stored in clear text inside DB. It is recommended to encrypt with a 3ds encryption algorithm and the secret

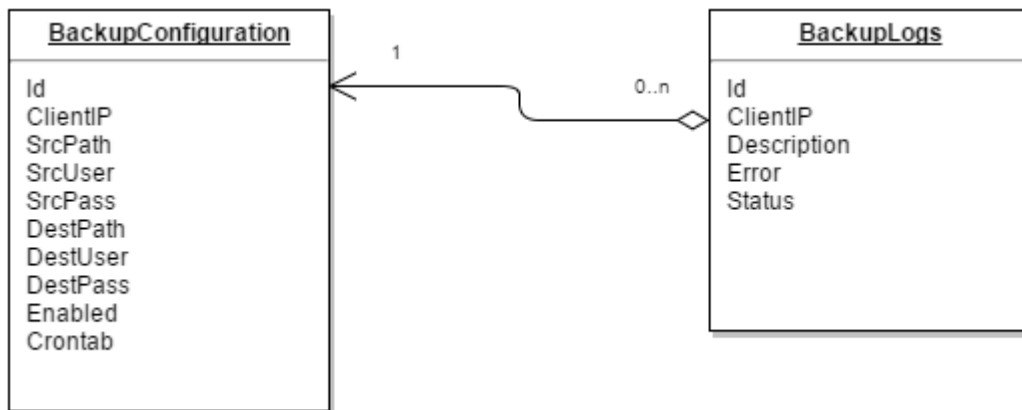
Field	Type	Description
		key to be shared between the apps that needs to decrypt encrypted field (e.g. have a global encryption key stored on WEB API that will be used by REST API when creating the record and later used by all agents when need to decrypt this field).
DestPath	(string)	Destination path on the LAN computer where agent will copy files and subfolders from SrcPath. Should be a valid UNC path (e.g. '\\NAS01\PROJSTORAGE\JOHN')
DestUser	(string)	User name credential needed when impersonating for gaining access to UNC shared folder on LAN computer.
DestPass	(string)	User password credential needed when impersonating for gaining access to UNC shared folder on LAN computer. (optional) – share security
Enabled	(boolean)	Specify if the configuration is enabled or disabled (when disabled the agent should not take it into account when evaluating the schedules)
Crontab	(string)	Crontab expression as detailed in https://en.wikipedia.org/wiki/Cron We choose this format because it is compact and fits widely with most scheduling libraries and engines available.

BackupLog entity model

Field	Type	Description
ID	(int) Identity Primary Key	Should be automatically generated and maintained by database store. Should be unique per collection.
ClientIP	(string) Index nullable	Represents the agent (computer) IP address and acts as a Foreign Key in relation with BackupConfiguration entity. It is needed when an agent is querying for its own backup logs.
ConfigurationID	(string) Index nullable	Represents the configuration ID and acts as a Foreign Key in relation with BackupConfiguration entity. It is needed when querying for a specific backup configuration activity in backup logs.
Description	(string)	The detailed description of the logged operation.
LogError	(string)	Error message in case there is an error that needs to be reported.
Status	(string)	Used to describe the status of the agent when reporting to log.
LogDateTime	(datetime)	Date timestamp of recorded operation

7.2 Database schema

The previous presented entities can be layout in the following relationship diagram



Entity models described earlier only cover the strict needs of storing normalised data. It is important to note that in a real scenario we should also take care of concurrency conflicts. For this reason each table will be enriched with a new field named **[RowVersion]** that will be a timestamp which will change (automatically by database) when a new update is performed against the record. This way any subsequent concurrent update will fail giving the opportunity to user/app to review the changes before re-submitting or discarding last changes.

8. User Interface – web application

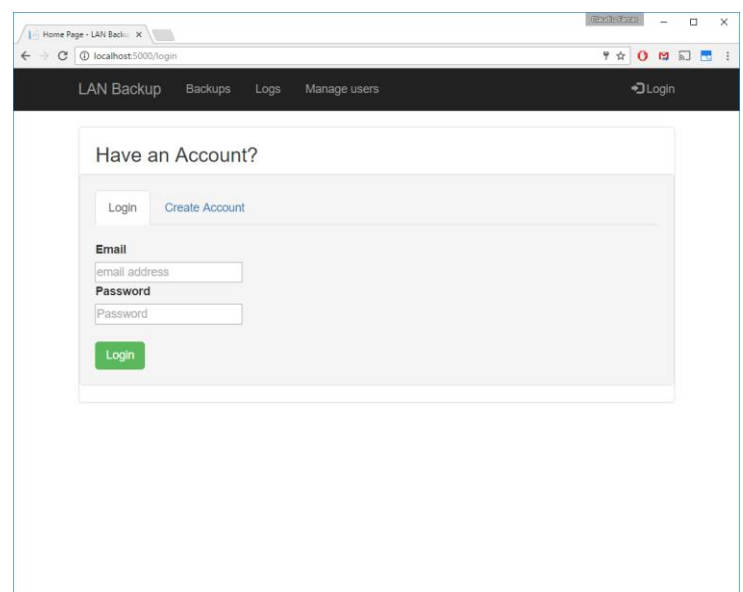
8.1 Login view

Fields in this view:

- Email address as user name
- Password

Actions in this view:

- Top navbar navigation
- [Log In] button – performs login authentication
- [Register] button – navigates to register view
- [Forgot your password] link – navigates to password recovery view



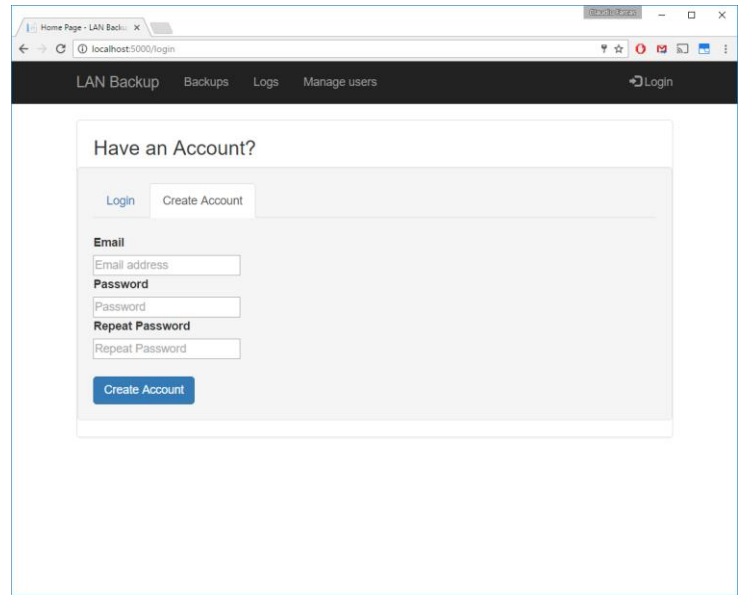
8.2 Register view

Fields in this view:

- Email address as user name
- Password
- Retype password

Actions in this view:

- Top navbar navigation
- [Register] button – navigates to register view



8.3 Dashboard (home) view – (Optional)

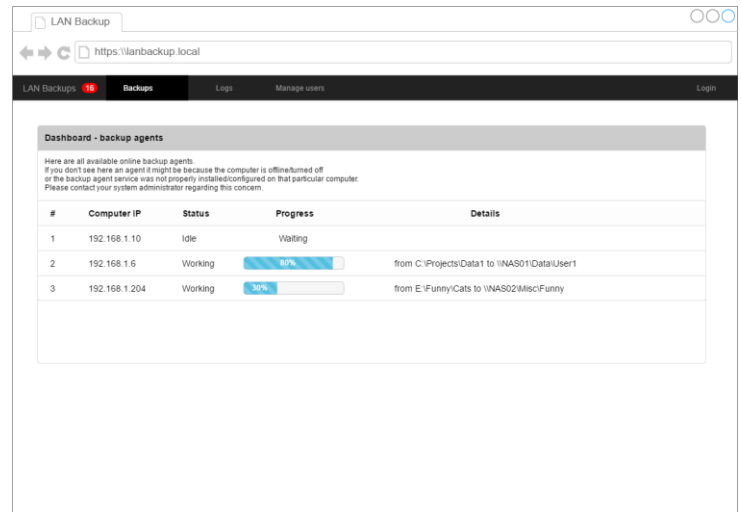
This view is an extra feature nice to have since the agents already communicate with WEB API services via SignalR.

Actions in this view:

- Top navbar navigation

Notes:

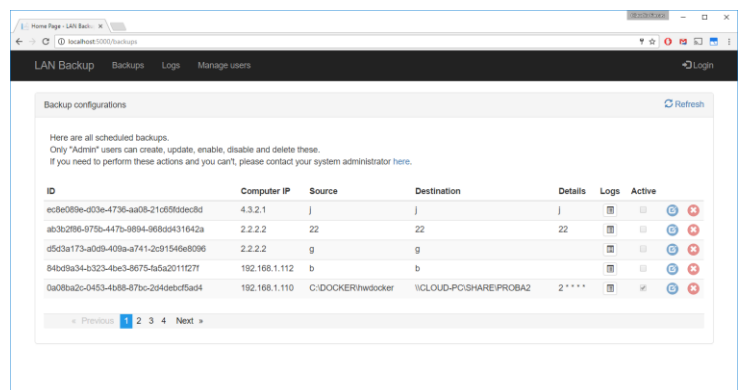
- The badge on the “LAN Backup” navigation button representing this page shows the number of currently working backup agents



8.4 Backups view

Actions in *list* view:

- Top navbar navigation
- [New configuration] button for adding new backup configs – visible only to “Admin” users - shows Create configuration popup.
- Each row has these buttons:
 - [logs] to view logs for selected configuration/selected computer IP
 - [active] to enable/disable the configuration (“Admin” only)
 - [edit] to edit the record (“Admin” only) – shows Edit configuration popup
 - [delete] to remove the record (“Admin” only). The deletion must be confirmed before taking place.



- List pagination buttons at the bottom of the list

Create backup configuration popup.

Fields:

- Client IP – should validate to be an IP address
- Source and destination paths, user and password pairs.
- Crontab expression with an interactive crontab picker.

Actions:

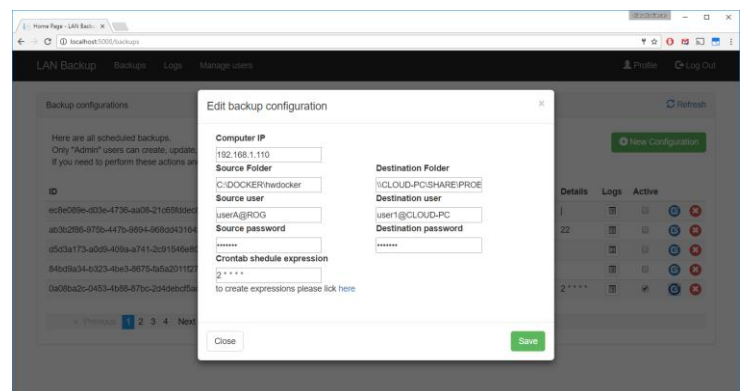
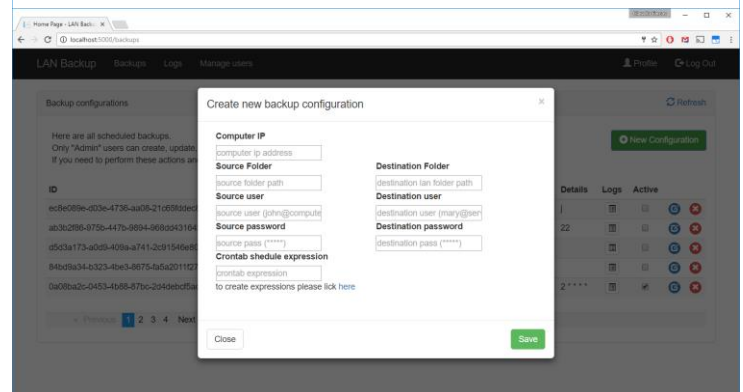
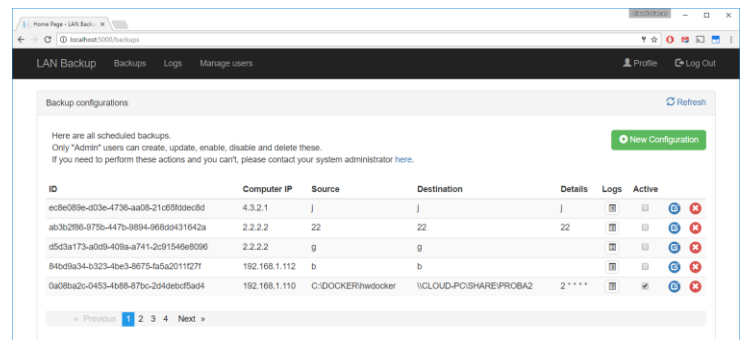
- [Save] button – creates and persist the configuration via REST API in the system.
- [Cancel] button

Edit backup configuration popup.

(very similar with creation popup)

Notes:

To make it different than creation we choose blue color for the title bar and panel border.

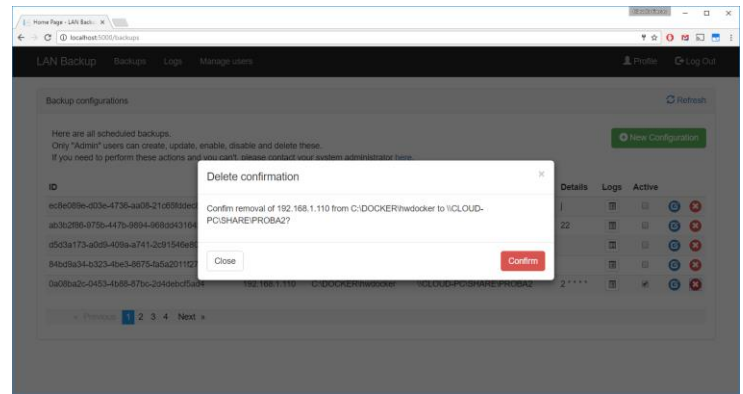


Delete backup configuration confirmation popup.

Only users from role “Admin” can delete a backup configuration.

If confirmed the delete will be performed via REST API and confirmed back to the user (the deleted row should disappear from the table – via a data refresh on the table).

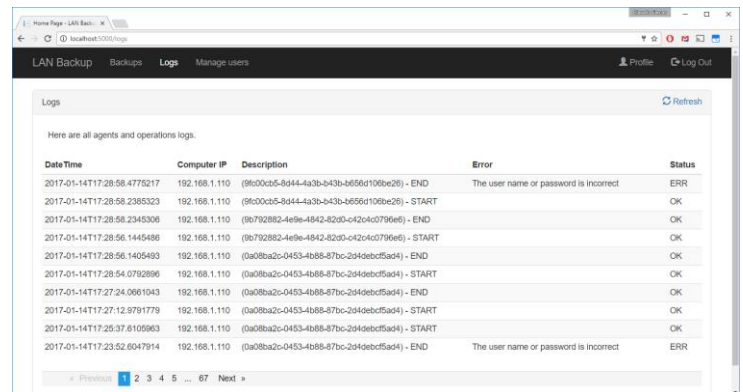
Only the backup configuration will be deleted. All logs generated by the associated agent with remain in database.



8.5 Backups logs

Actions in this view:

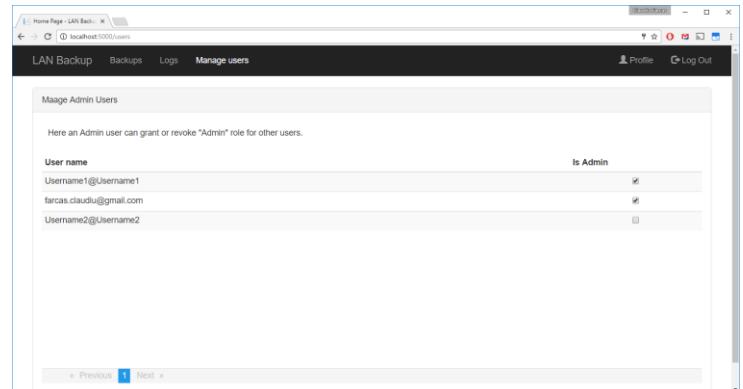
- Top navbar navigation
- List pagination buttons at the bottom of the list



8.6 Users management view

Actions in this view:

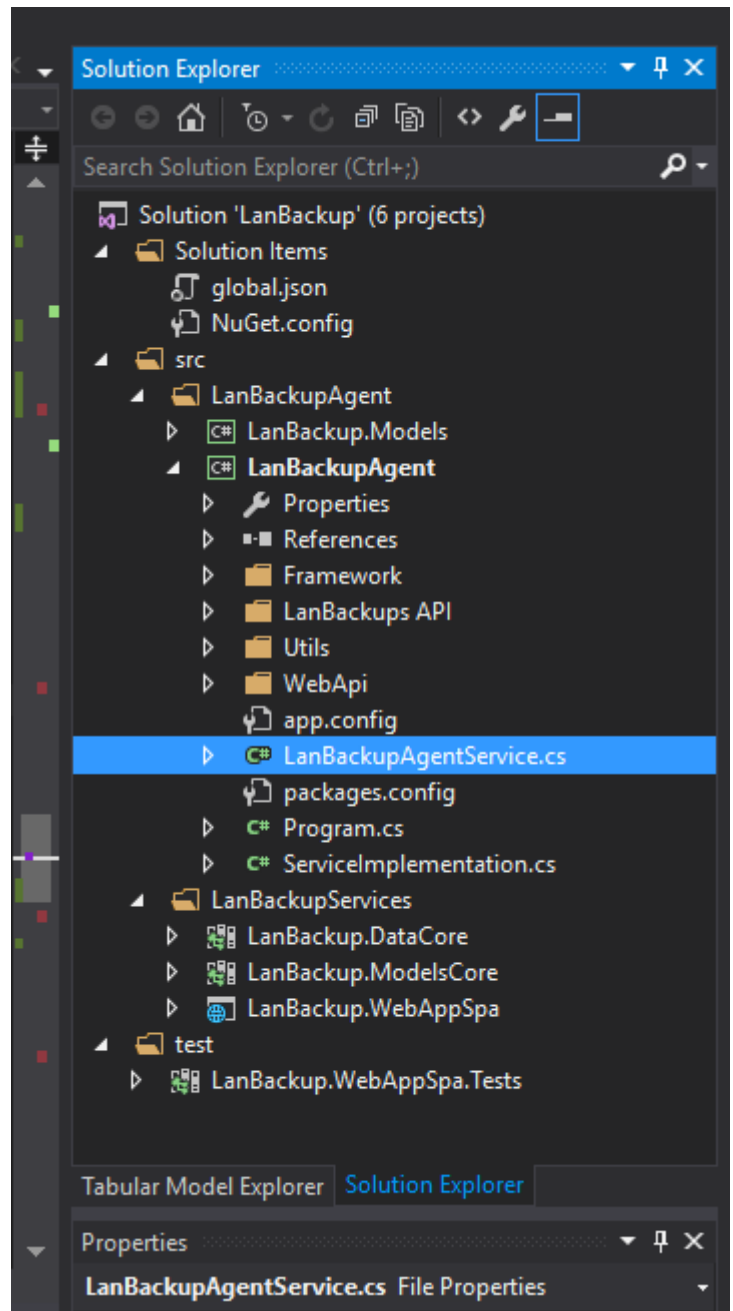
- Top navbar navigation
- Each user record has a toggle button to promote to or demote from “Admin” role
- List pagination buttons at the bottom of the list



9. Software project structure

The solution structure as follow:

- LanBackupAgent – a windows service residing on the client computers
- Web API and Web Application projects
- Testing projects



Server vs Client

Unit testing and code coverage

There are defined several unit tests and integration tests in the WebAppSpa.Test project covering the web api and MVC controllers. Code coverage is pretty low at this stage.

Run **dotnet tests** in in test project for rapid roll of the tests. Please bear in mind that integration tests can take some time as they will spin an entire MSSQL local instance.

10. System security

- Data input validation
- Communication infrastructure (HTTPS recommended)
- server isolation
- implementing an upgraded authorization mechanism (like OAuth 2 steps validation).
- backup agents should at least authenticate with a basic auth schema.

11. Deployment and Configuration

11.1 Server

The LanBack.WebAppSpa is the project from where we can publish the solution in Azure cloud or make it a package or directly into a folder that can be installed manually into any supported deployment architecture (IIS, docker, linux – the web app is dot net core, so it should run without issues).

After installation needs to be configured **appsettings.json** file to supply 2 connection strings for two mssql databases: one for storing identity credentials and another to store application data presented.

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=aspnet-LanBackup.WebAppSpa-41-8000-1e69f75536f7;Trusted_Connection=True;MultipleActiveResultSe",
    "BackupsConnectionString": "Server=(localdb)\\mssqllocaldb;Database=BackupsLAN;Trusted_Connect",
  },
  "FileLogging": {
    "Path": "logs/lanbackup_{Date}.log"
  },
  "Logging": {
    "IncludeScopes": false,
  }
}
```

11.2 Client

On the backup agent side, these are windows service and will run only on windows machines. I will advise building an installer that can be easily deployed into a corporate network.

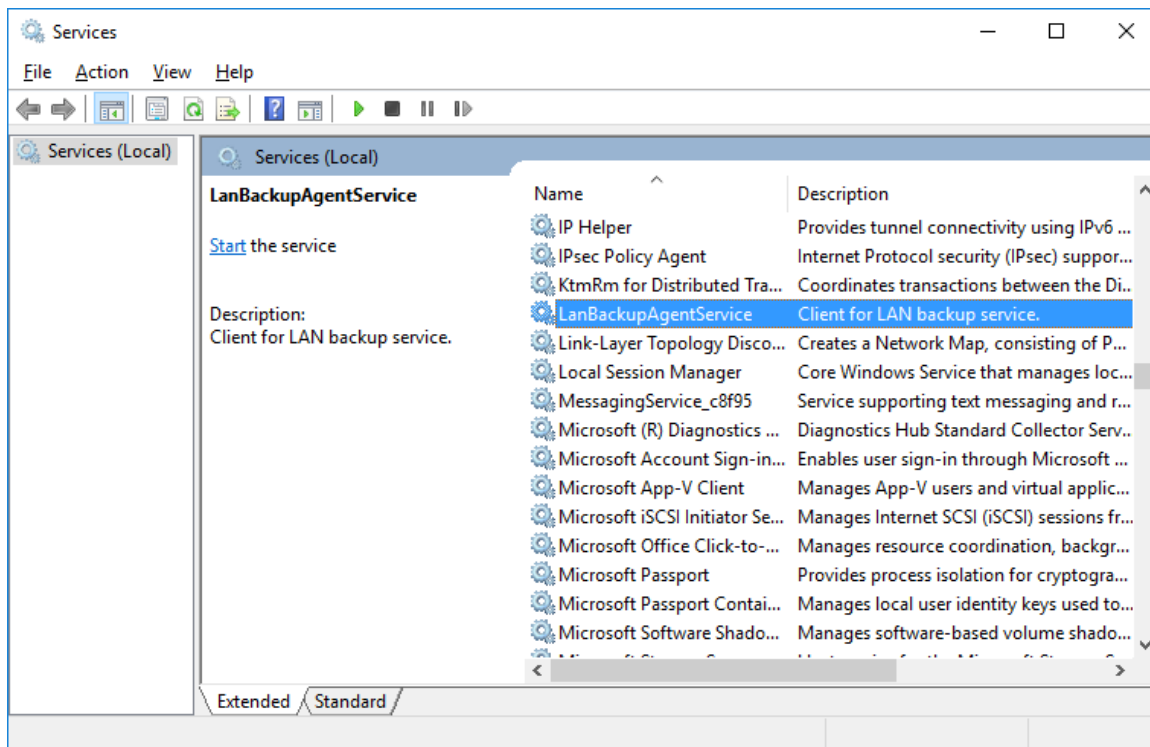
Another simple way to install is just to take the output bin folder (release) and copy it where it needs to be installed. There, we can issue the **install** command to install it as a service:

LanBackupAgent.exe -install

You can always uninstall it with

LanBackupAgent.exe -uninstall

Bellow showing how it should show up in the **services.msc** administration console:



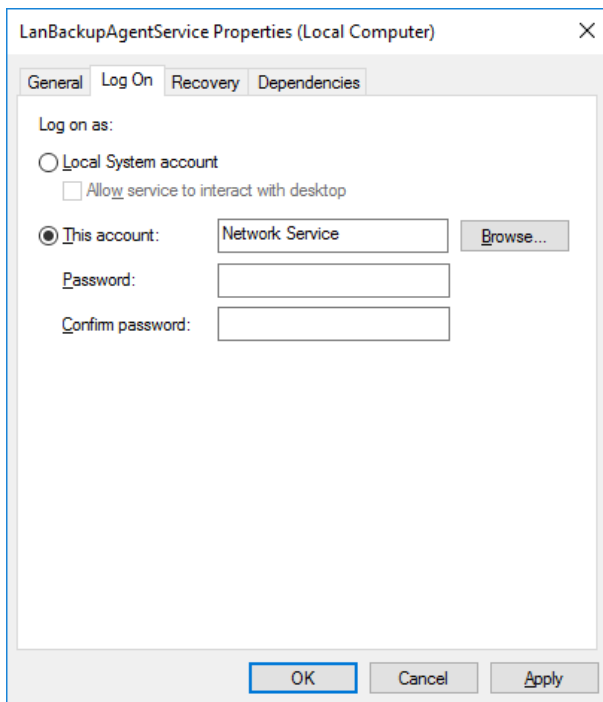
After installation in the folder there is a **LanBackupAgent.exe.config** configuration file that can be edited to provide webapi network address.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <configuration>
3  <appSettings>
4    <add key="webApiUrl" value="http://localhost:5000" />
5  </appSettings>
6  <connectionStrings>
7  </connectionStrings>
8  <runtime>
9    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1"
10      <dependentAssembly>
11        <assemblyIdentity name="Newtonsoft.Json" publicKeyTo

```

After we need to set a security setting from services.msc console, right click over the service and configure the LogOn user to be **NETWORK-SERVICES** (this is IMPORTANT in order for service to be able to copy files over network):



And lastly font forget to set the startup type to **Automatic** and even to give it a start.

In case of issue during functioning, the service emits a **log.txt** file in the same installation folder that have precious information needed to clarify the issue.

Summary

The product should be simple to install and use.