Oscar Nordman

Piloting and Implementation of Printing Service

Metropolia University of Applied Sciences

Bachelor of Engineering

Electronics

Bachelor's Thesis

9 May 2025

Abstract

Author:	Oscar Nordman
Title:	Piloting and Implementation of Printing Service
Number of Pages:	36 pages
Date:	9 May 2025
Degree:	Bachelor of Engineering
Degree Programme:	Electrical and Automation Engineering
Professional Major:	Electronics
Supervisors:	Teemu Loman, CTE-team manager Tatu Suomi, Senior Lecturer

---

This thesis covers the phases of upgrading existing product label printing system to a modern REST API style one. Demand for this project arose as the old label printing system, Bartender 2016, was starting to get a bit outdated and official support was already discontinued from April 2023 onwards. Optional label printing environment was also studied over a year ago by another party, but the result was that the current Bartender system was to be updated.

In chapter two the focus is on the overview of the key systems within the Bartender environment. Bartender Designer, the label template design tool is introduced, the main system console Administrator Console is discussed, and the label template database Librarian and its corresponding web service Print Portal are explained. Basics of REST API and different versions of used RESTful designs within the Bartender are also introduced.

In chapter three the testing environment used for study purposes is introduced. It was created to gather as much data as possible and to choose the best options out of three REST APIs available in Bartender 2022. Also, all three REST APIs are showcased and compared with one another.

The last part of this thesis consists of the initial testing results, which looked promising, and discussion about the implementation process of the whole system. Some improvements directed to the logging of the printing jobs was already recognized and they will need some attention in near future. Overall, the project was a success and the initial roll-out readiness for production use was achieved.

Keywords: Label printing, labels, REST API, Bartender

Tiivistelmä

Tekijä:	Oscar Nordman
Otsikko:	Printing Service pilotointi ja implementointi
Sivumäärä:	36 sivua
Aika:	9.5.2025
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikan tutkinto-ohjelma
Ammatillinen pääaine:	Elektroniikka
Ohjaajat:	CTE tiimimanageri Teemu Loman Lehtori Tatu Suomi

Tämä opinnäytetyö kattaa tuotetarrojen tulostamiseen tarkoitetun järjestelmän uudistamisen vaiheet REST API tyyliseen moderniin järjestelmään. Tarve kyseiselle projektille tuli vanhan Bartender 2016 systeemin vanhentuessa ja virallisen tuen päätyttyä jo huhtikuussa 2023. Vaihtoehtoisia tulostusympäristöjä tutkittiin jo toisen projektin toimesta reilu vuosi sitten, mutta kyseisen projektin lopputulos oli pidättäytyä nykyisessä palveluntarjoajassa Bartenderissä ja päivittää se uusimpaan versioon.

Kappaleessa kaksi käydään läpi Bartenderin pääjärjestelmät eli Bartender Designer, jota käytetään tarrapohjien suunnitteluun, Administrator Console, joka toimii järjestelmän hallintakonsolina, sekä tarrapohjien tietokantana toimiva Librarian sekä sen verkkopalvelu Print Portal. Kappaleessa esitellään myös REST API:n perusteet sekä Bartenderin tarjoamat REST API vaihtoehdot.

Kolmannen kappaleen tarkoitus on esitellä käytetty testausympäristö, jonka avulla kerättiin ja analysoitiin dataa mikä kolmesta Bartender 2022 tarjoamasta REST API arkkitehtuurista toimii parhaiten tarkoitetussa käyttöympäristössä. Kappale myös esittelee nämä kolme eri vaihtoehtoa sekä vertailee niitä keskenään.

Viimeisessä osiossa käsitellään alustavia testaustuloksia sekä keskustellaan koko järjestelmän implementointi prosessista. Joitain parannusehdotuksia löydettiin jo testausvaiheessa, jotka tarvitsevat huomiota lähitulevaisuudessa, mutta eivät kuitenkaan estäneet uuden järjestelmän tuotantoon ajoa. Kokonaisuudessaan projekti oli onnistunut ja haluttu tuotantokypsyys saavutettiin.

Avainsanat: Tarratulostus, tarrat, REST API, Bartender

Contents

List of Abbreviations

.NET: Open-source platform released by Microsoft and used for building desktop, web and mobile applications that run on various operating systems.

API: Application Programming Interface. Way of communication between multiple computer programs or components with each other.

REST: Representational State Transfer. Software architectural style on how an API should work.

SDK: Software Development Kit. Collection of software development tools designed to assist software developers in creating applications for specific platforms.

SSO: Single Sign-On. Authentication scheme that allows a user to log in with single ID to several related, yet independent software systems.

URL: Uniform Resource Locator. Indicates the location of a resource and the protocol used to access it.

1 Introduction

The purpose of this bachelor's thesis work was to study, test and pilot a new label printing architecture to replace an old unsupported system. The main tools and software remained the same as before, but the way of handling and maintaining data related to the label printing and the label templates were redesigned to fully support RESTful API architecture. This included updating the software version of the used label printing environment called Bartender, optimizing the usage of the label printing tools, setting up a centralised database for the label templates, standardizing the label source data fields and implementing the REST API queries to the C# based test platform.

1.1 Leading to This Project

Demand for this specific project arose from multiple sources. Firstly, the tools used in label printing had reached their end-of-life, meaning that within the time of this project, the official support for the Bartender software would not be valid [1]. This means that new licenses for the old Bartender 2016 used could not be bought, and there was no guaranteed support or compatibility e.g. for Windows 11 in the future. Secondly, the workflow of the label design and printing was disorganized and hard to manage, due to the fact that label templates were not centralized anywhere. Location of the templates varied, label names and data fields were not standardized in any way, plus the number of different labels used made the whole system very disorganized. Thirdly, Vaisala is currently focusing in modernizing its way of operating meaning many fields within the company are getting a "facelift" label printing included.

The idea of renewing the label printing started just over year ago, when a short study of competing label printing product NiceLabel and its features was done. The conclusion of the study was that upgrading the already known system was a better and more cost-effective option, leading to this Printing Service project.

1.2 Starting Point

Workflow of the label printing before the Printing Service implementation could be roughly illustrated as seen in the Figure 1. A test station runs the test software, which communicates with an installed Bartender 2016 software via a build in .NET application. Before printing, Bartender software will load and open the label template in the background according to a given file path. This path varied from local hard drive to shared virtual drives within the company network. Label data was then sent from the test software to Bartender, which filled preconfigured named data sources holding slots for information such as serial number, order code or product name. Label templates usually held other information too, such as printing options, printer used etc. When all the wanted data fields were filled, Bartender would take care of the printing action by sending the command to the printer queue of the chosen printer. Usually, this printer would be pre-saved in the label template as printing settings as mentioned before. If the said printer would not be available, Bartender could use another printer based on installed printers in the test station PC.

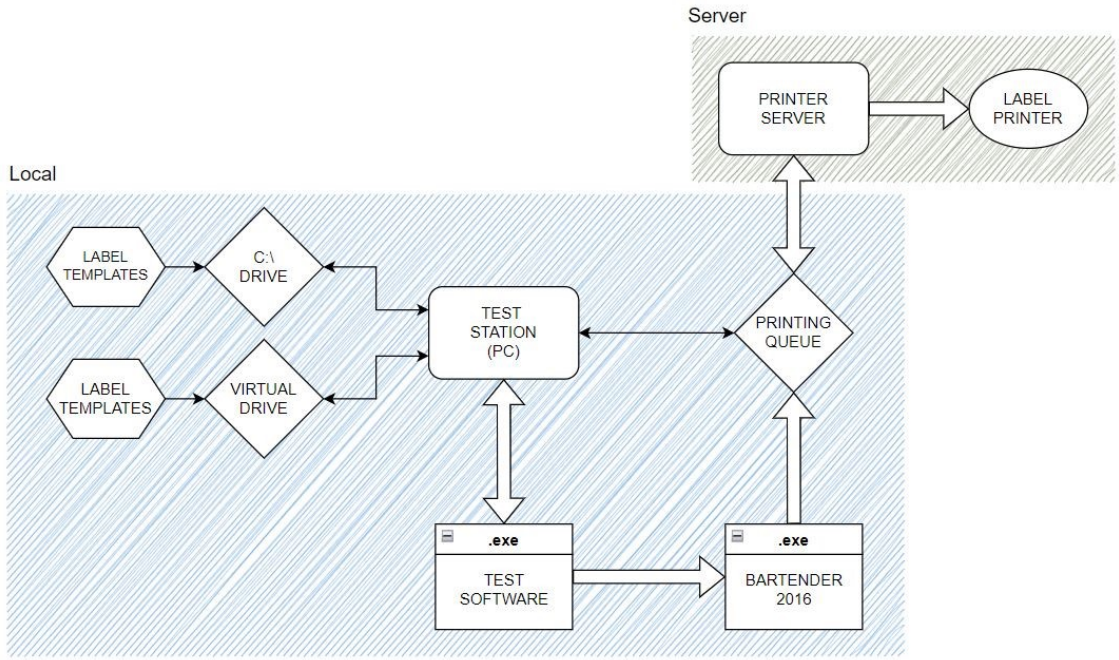

Figure 1. Dataflow of the old way of label printing.

The problem with this method of printing was that everything happened locally. Label settings were set up for a specific printer and software version, there was

no footprint of what data had been sent to the label or how it was handled. Making changes to the label templates had to be very often made locally, or the label settings would not match the printer – software combination. This basically caused station downtime, which was not an optimal way of working.

1.3 Vision

The new Printing Service would be web application based, meaning more data would be available for more endpoints, and in a much simplistic manner. At the test stations, there would be no need for a physical installation of the Bartender software, meaning also that the test software would not need a build in Bartender .NET application. Ideally, the only Bartender installations would be located on the Printing Service server and on developer's PCs doing the label design work. All the label templates would be in a single database, making them easier to access and manage. In test station design point of view, label templates would have standardized names and standardized data fields, unifying the way label data is handled between different test stations. All the printers would be installed in a single server, and the test stations would only need to know which specific printing queue to call for. The original Printing Service concept image can be seen in the Figure 2 below. [2.]

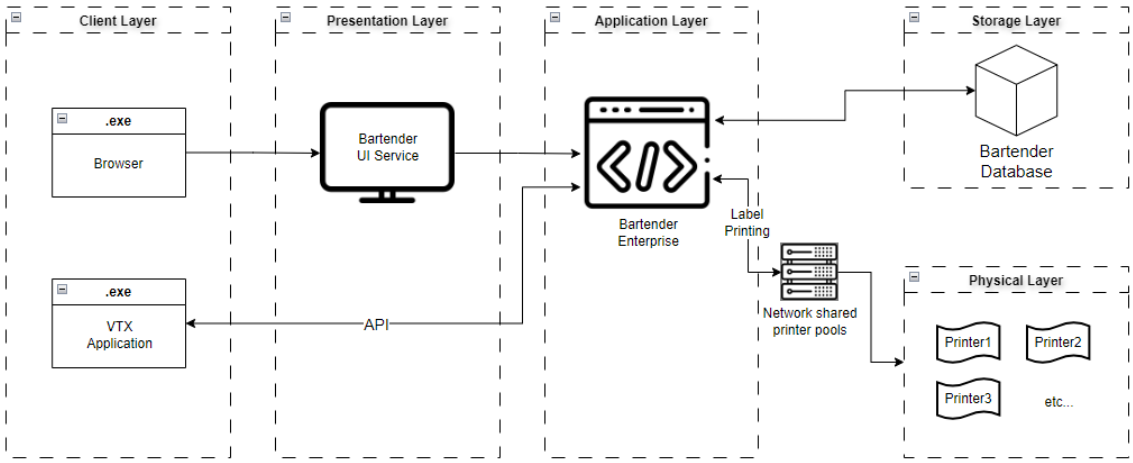

Figure 2. Concept image of the Printing Service project.

2 Bartender 2022 Enterprise

This thesis work was mainly focused on Bartender environment and its new features regarding label printing and management. Bartender 2022 Enterprise is the newest and most comprehensive version of the said software and provides multiple new features compared to the old 2016 version. Bartender is available with 4 different editions; Starter, Professional, Automation and Enterprise, first one mentioned being the most basic one and the last with full features available. Different editions and their features are also presented in the Figure 3. [3, p.10.]

In short, Bartender 2022 Enterprise is a label management environment with broad range of features that make label managing and printing secure and easy. The most essential ones are listed and presented in this chapter.

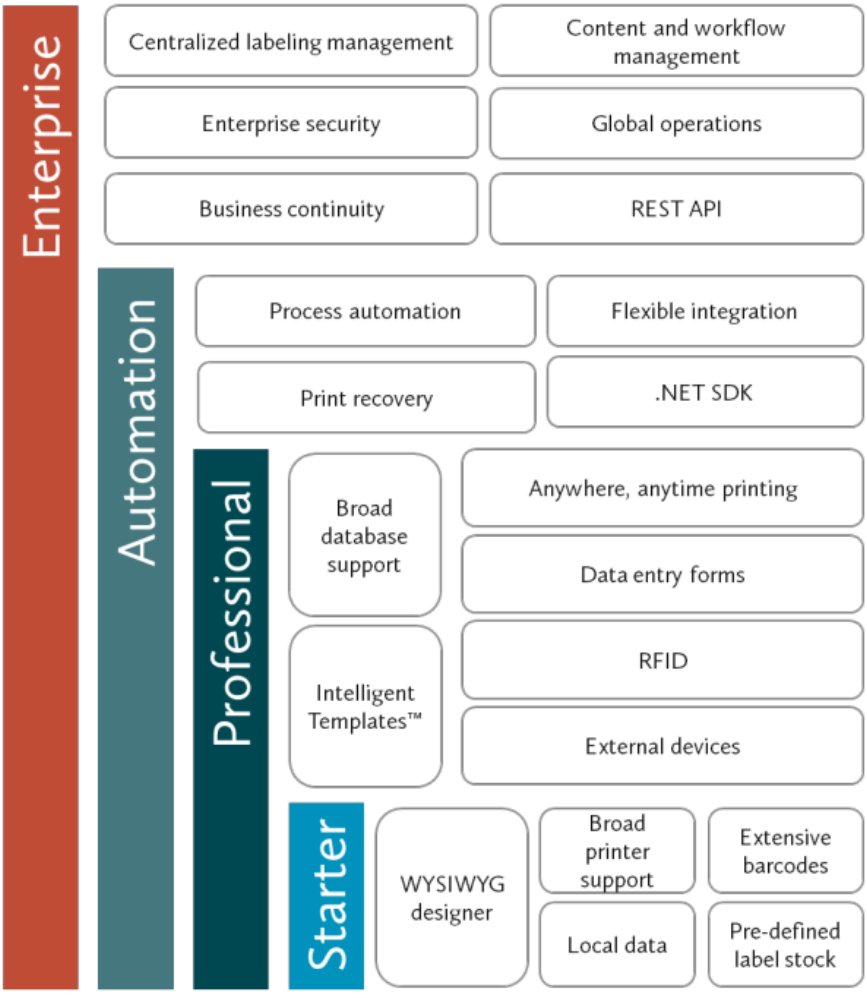

Figure 3. Bartender 2022 editions and their features. [3, p.10]

2.1 Bartender Designer

As the name suggests, the Bartender Designer is the designer software for creating professional labels. It consists of multiple tools helping in adding standardized text, pictures, shapes, barcodes, data matrixes, etc. to the label design. It also supports different types of data fields, that can be used to populate certain parts of the label. These data fields are either global or named data fields. Global fields are common to all the templates within the Bartender database, meaning the data is shared between all the labels. Named data fields are more dynamic. They reserve a defined data slot within the label, which then can be populated by external data. [3, p.7.]

While designing the label template, printer, printing settings, and used label material can be pre-defined to the label data for better consistency and accurate label dimensions. These settings consist of for example printing darkness, printing speed and image processing options. Set values are saved in the template and used when printing, if not modified during the printing process.

2.2 Administration Console

The Administration Console serves as a centralized hub, where the administrators have the ability to manage and adjust system settings that affect all applications in the Bartender Suite. A wide range of tasks can be done within the console, from setting up a shared database to overseeing Bartender licenses and printer usage. All the controllable modules are [4.],

- Printer Setup
- File Locations
- Database Connections
- Integrations
- Security
- Licensing
- System Database
- Email Servers

- Librarian Setup
- Windows Services.

2.3 Librarian

Bartender Librarian is an application within Bartender ecosystem that allows groups of designers, administrators and other users to co-operatively and securely manage how Bartender documents and other related files are stored and revised. All Bartender related files are stored in a library folder, which is a dedicated set of tables in the Bartender System database. This centralized file storage location allows for easy tracking and management of its contents. Access and permissions to the files can be controlled via Administration Console.

Librarian also provides revision control, retaining copies of all previous version of a document. This allows the documents to have a history in which all the changes are saved and stored. Who, when, and what was altered can be seen and copies of previous iterations can also be retrieved or reverted to. When a file is managed via Librarian, it is also checked-in and locked to that user, making it impossible for multiple users to modify the same file simultaneously. This prevents overwriting other user's changes. [5, p.6-10.]

Using Librarian also allows for use of workflow management. All the Bartender files within the library using a workflow, have pre-defined states and all the states can have their own set of rules, for example who can change the current state or is the said state in production or not. Bartender workflow itself is an object based graphical drag and drop designer. Creating custom workflows is easy, and different libraries or directories can have their own workflows. It is also possible to configure specific recipients to get an automatic email notification when files move to a certain state, for example in review state. [5, p.11-13.]

2.4 Print Portal

Bartender Print Portal is a web-based application that provides an interface to select and print Bartender documents from any computer connected to the main

system server. It can be integrated with Librarian, making it quick and easy to access desired documents. When integrated, Bartender documents can be searched with a keyword, can be grouped and sorted, can be previewed and their revision history and comments can be seen, and workflow states can be updated.

Print Portal must be installed on the used system server along with Bartender. It supports all web browsers and all versions of Windows which are supported by Microsoft. To access Print Portal, authentication can be enabled, making it more secure from anonymous access. Either user authentication with a log in screen or Single Sign-On (SSO) can be used. SSO allows access to multiple applications and websites with just a single use of credentials, while user authentication gives access to only Print Portal application. In short, Print Portal is a web-based Librarian with limited features. [6, p.4-16.]

2.5 REST API

REST (Representational State Transfer) is an architectural style for designing network applications. A REST API (Application Programming Interface) is a set of rules for how a client should request data, and how a server should respond to those requests. The main rules are:

- **Uniform Interface:** The method of communication between a client and a server must be uniform and consistent, simplifying the architecture and improving visibility of interactions.

- **Client-server decoupling:** The client and the server are separate entities that communicate over a network. The client is responsible for the user interface, and user experience, and the server is responsible for processing requests, performing operations, and storing data.

- **Statelessness:** REST APIs operate on a stateless basis, which implies that every request must carry all the required data for its execution. This means that REST APIs do not depend on sessions stored on the server

side. It is not permissible for server applications to save any information related to a client's request.

- **Cacheability:** Client or server can cache responses. Responses must contain information if it is cacheable or not. This is to prevent clients reusing outdated or inappropriate data in response to further requests.

REST APIs typically communicate over HTTP using methods such as GET, POST, PUT, DELETE, etc. They can return data in different formats, JSON being the most common. REST APIs are used for example in web and mobile services. They allow different software systems to communicate with each other, even if they are written in different programming languages, or running on different platforms. [7.]

2.5.1 Bartender REST API

The Bartender REST API feature was introduced in Bartender 2022. It allows automation of printing jobs over a network using six different REST commands. These calls can be used to run a variety of actions that are used in the Bartender Actions API, which include actions from Bartender Designer, Integration Builder and in Process Builder. The main categories of the said actions are:

- Print Actions – Print document, command script or BTXML script.
- Input Actions – Read a file, listen to serial port etc.
- Output Actions – Write message to log, send web service request etc.
- Execute Actions – Set variable, run a shell command etc.
- File Actions – Create a folder, copy a file, delete a file etc.
- Transform Actions – Search and replace, search and delete etc.
- Database Actions – Execute SQL, for each database record etc.

To make Bartender execute various actions, POST method is used. Each POST command needs to include which action is used and its variables. After sending the command, it is assigned a unique ID. After the POST command is

successfully completed, it can be queried with a GET method using the same identification. The same applies for the PATCH method, which requires a known ID. The only difference is that with PATCH, ongoing script properties can be changed or cancelled altogether. List of submitted commands and their status can be seen with GET method without any ID included. The REST API requests of the Bartender Actions API can be seen in the Table 1 below. [8.]

Table 1. Bartender Actions API endpoints [8].

HTTP Method	URL Path	Description
POST	/api/actions	To submit a script to run in the server
GET	/api/actions/{id}	To get the status and messages of a running script
PATCH	/api/actions/{id}	To cancel or change properties of the script
GET	/api/actions	To get a list of scripts that were submitted to the server
GET	/api/actions/{id}/variables	Retrieves the variables associated with the script
GET	/api/actions/{id}/variables/{VariableName}	Retrieves the value of a particular variable associated with the script

2.5.2 Print Portal REST API

Bartender 2021 and after also supports the Print Portal REST API. It consists of a total of four endpoints, which include methods for authenticating, listing existing libraries, labels and printers, and printing itself. This is done by issuing queries via GET and POST methods. Results of the queries are retrieved in the format of JSON. A clear difference from the Bartender REST API is the way POST methods work. A single POST endpoint takes care of just a single action, making its usage

simple and easy. By comparison, in the Bartender REST API, a single POST method could be configured to do any action or array of actions within the Actions API limit. The benefit of using Print Portal REST API as already mentioned is its simplicity and ease of use, but the clear drawback is the limited number of commands available that can be seen in the Table 2 below. [9.]

Table 2. Print Portal REST API endpoints [9].

Endpoint	HTTP Method	Description
Authentication	POST	Using username and password to authenticate for future requests. *[/api/v1/Authenticate]*
AutomatedPrint	POST	Performs an automated print job to the specified file path inside a given library. *[/api/v1/print]*
Library	GET	Returns the available libraries. *[/api/v1/libraries/{id}]*
Printers	GET	Returns the available printers. *[/api/v1/printers/{printerName}]*

*optional more detailed query

2.5.3 Integration Builder

Integration Builder is a tool that allows for creation and testing of custom integrations with multiple trigger options. The same actions of Bartender Actions API are available to use, as listed in the chapter of the Bartender REST API. As a reminder, these included Print Actions, Input Actions, etc. In the sense of a web service requests, this basically means that custom endpoints can be created. So, in architectural point of view, Integration Builder can be thought as a hybrid of Bartender REST API and Print Portal REST API. A single endpoint does only a specified action(s), however they are fully customizable with Actions API tasks. Every single used task needs to be configured separately, and the order in which they are performed can be specified.

Use of variables is also supported, and some actions automatically save data to predefined variables, such as EventData or Response. Other variables depending on the action can contain for example some file path, name of a printer used etc. Variables are also fully customizable in their name and content they hold. Variables can also be defined as global or integration specific. The latter are only available inside the single integration, and global are shared among all the created integrations used.

Web service request is only one of the integration options in Integration Builder. The others are file, database, email, network socket, serial port, message queue and time schedule integrations. Used actions remain the same, only the trigger event is different with each integration option. For example, with a time schedule integration, predefined action(s) will run, when they are scheduled to run, or with database integration actions are triggered with modification of some specified database. [10.]

3 Test Setup

A flowchart of the test setup can be seen in the Figure 4. Basically, everything revolves around the Bartender test server. It functions as the licencing server for the whole system, it has the direct connection to the centralized database on another server, it has all the used printers and their drivers installed, and it is the heart of the Bartender REST API and Print Portal. As it can be seen, the Bartender test server is somewhat isolated and can only be accessed indirectly via some API. The thick white lines represent the test station connection to the main server via Bartender REST API. The dashed lines represent the connection to the Print Portal REST API. Finally, the solid thin black lines represent the Bartender 2022 system connections. Bartender installation presented as Bartender 2022 box, includes all the Bartender Enterprise installed software’s such as Bartender Designer and Administration Console.

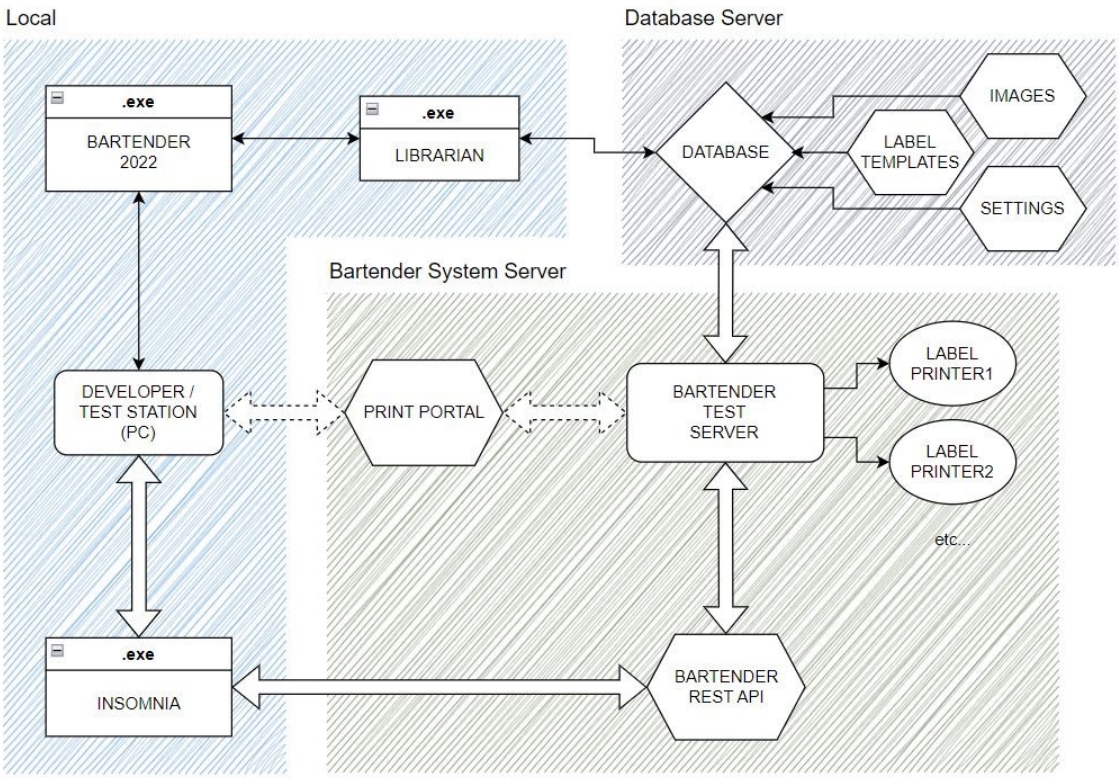

Figure 4. Flowchart of the test setup.

The Idea of the test setup is to simulate the real-life test station and developer behaviour at the same time, while the new system is up and running. On the final implementation, test stations will not have the Bartender 2022 installation, and the REST API requests are sent automatically via an implemented subclass in the test software platform, and not by some third-party software like Insomnia used in this testing case. The reason for the Bartender installation in the testing environment, is to have access to all the same tools as a normal developer would. This means having quick access to all the Bartender system settings and to be able to play with the label templates and their settings with the Designer tool.

3.1 License Server

Licensing model of the Bartender Suite is very dynamic. The number of needed licenses is defined by the amount of label printers used. When issuing a printing command, the license server is contacted, and a single license is reserved from the license pool for the used printer. If the said printer is idle for over seven days, the reserved license is released. Bartender installation itself does not reserve a

license, even though contacting the license server with several actions is mandatory to check the Bartender activation status. [11.]

Licensing of the Bartender Suite is dealt by SLS (Seagull License Server), which tracks the used licenses. SLS is set up during the first Bartender installation, where a product key is used to activate the system. During the setup, location of the license server, and the sharing of the license with other computers in the same network is configured. Sharing the license is only needed if multiple Bartender installations are present, as in Vaisala's case. In this piloting project the Bartender environment and SLS was installed in a test server reserved for the whole experiment seen in the **Error! Reference source not found.** as "Bartender Test Server". Used licenses and their status can be tracked in Administration Console. It is also possible to relocate the whole license server using the Administration Console. [11.]

3.2 System Database

The Bartender System Database was created in a separate server with an installation wizard. Installation process was easy as the database itself was created automatically, only the location, connection and database maintenance settings needed to be manually configured. The system database acts as the central data storage for all the Bartender applications and is directly connected to the System Service located on the Bartender test server, the location of which can be seen in the **Error! Reference source not found.**.

Database stores the following data [12, p.3]:

- events encountered by any Bartender applications
- information about each print job that is sent from Bartender
- security settings and permission checks that are defined in Administration Console
- template designs and preview images
- global data fields

- librarian files and revision information.

The created database for this project used centralized Microsoft SQL Server Express. Other options were also available, but Microsoft SQL was chosen as it is already used and is a familiar environment in Vaisala. The point of the centralized database is to have all the needed data in one location. This makes the data easy to maintain and access and minimizes the data redundancy. Downside for this data management strategy is that it is prone to higher data traffic, and risk of losing entire data in case of system failure is substantial. [12, p.3].

3.3 Labels

All the used labels were placed in a centralized location called Librarian. Its structure can be defined in multiple ways, but in Vaisala's case the idea was to separate label templates that are in production from templates that are still in the design process. Hence, the library structure seen in the Figure 5. Like the name suggests, the LabelProduction folder is meant for templates that are in production. This folder structure is supposed to be open and visible to all Vaisala personnel. The library called Pictures was created to hold items and pictures that might be added to the label while printing, for example subcontractor logos or special markings. The remaining libraries are meant for the design and development teams, that create the new label templates for the new products or modifies the existing ones. The design/developer team folders are to be hidden from most of the company and have strict access to them. All the libraries will use the same workflow once it is taken into use.

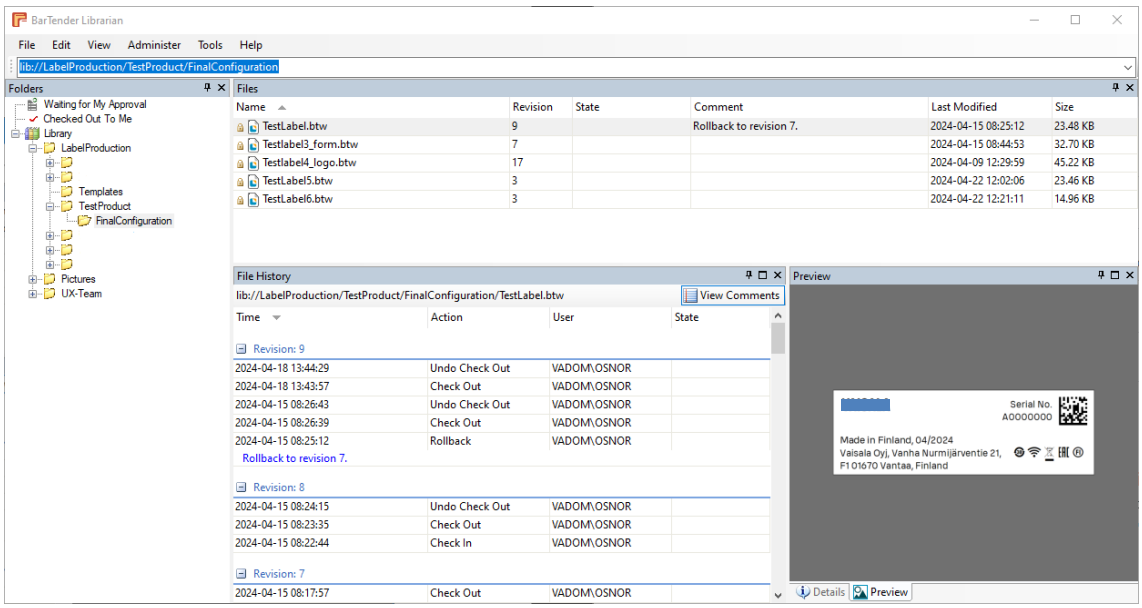

Figure 5. View from the Librarian.

When a template is opened from the Librarian, it is reserved for that specific user and the template is automatically opened in the Designer tool as seen in the Figure 6. On this view, the template is located in the middle of the screen and the assigned data sources can also be made visible as they are in the Figure 6. All the pre-defined named data sources are located on the left side and are intended to be uniform between all the future labels. The data sources are stored in the Bartender database and contains information like the example output, data type, and some transform settings like number of minimum and maximum characters allowed. The named data sources can just be dragged and dropped to an empty space in the template or dropped and automatically linked to an existing object. On my test environment, these data sources are not yet fully completed but are sufficient for testing purposes. Adding more or refining the old data sources is easy.

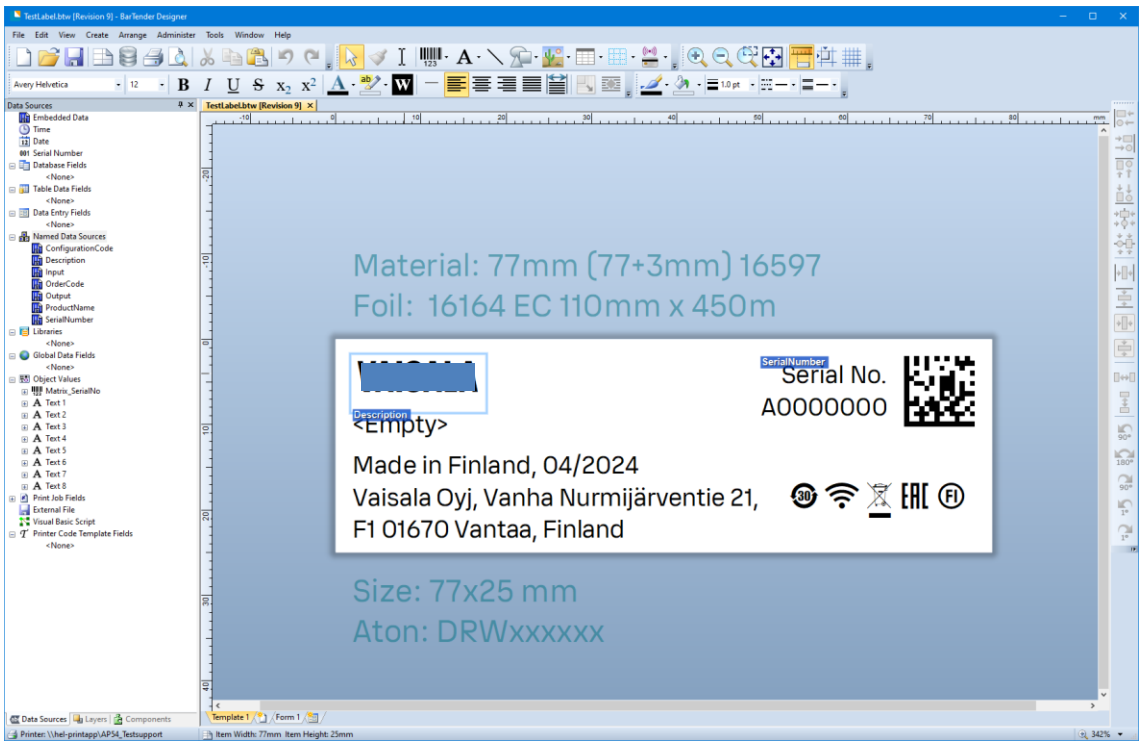

Figure 6. View of the Designer tool and an example template.

Librarian also supports usage of workflows to streamline the document creation process and to monitor a document's journey from its initial draft to publication. Workflow contains the procedure of generating files, overseeing reviews, and managing modifications until the file is deployed or published. Workflow can be fully tailored to the need by using the Librarian Workflow Setup dialogue. This includes defining the necessary approval stages and identifying the users who have the authority to approve or advance each stage. Also, documentation visibility can be managed via workflow states. [13, p.3.]

Below in the Figure 7, an early implementation of the workflow that was created can be seen. All the diamond shaped states are review blocks, which need approval of a specified personnel for the document to advance to the next possible state. Rectangular shaped states do not need any approval but depending on the phase they might force the user to write a comment or leave their digital signature behind. The idea of the process in Figure 7 is, that a new document goes to prioritization, where it is decided which team starts working on it. After the design work is finished, it is reviewed and taken into production. The documents that are in production can be retired, or an update can be requested

by anyone if a fault or improvement is detected. From there, the document goes back into prioritization and the whole process starts over.

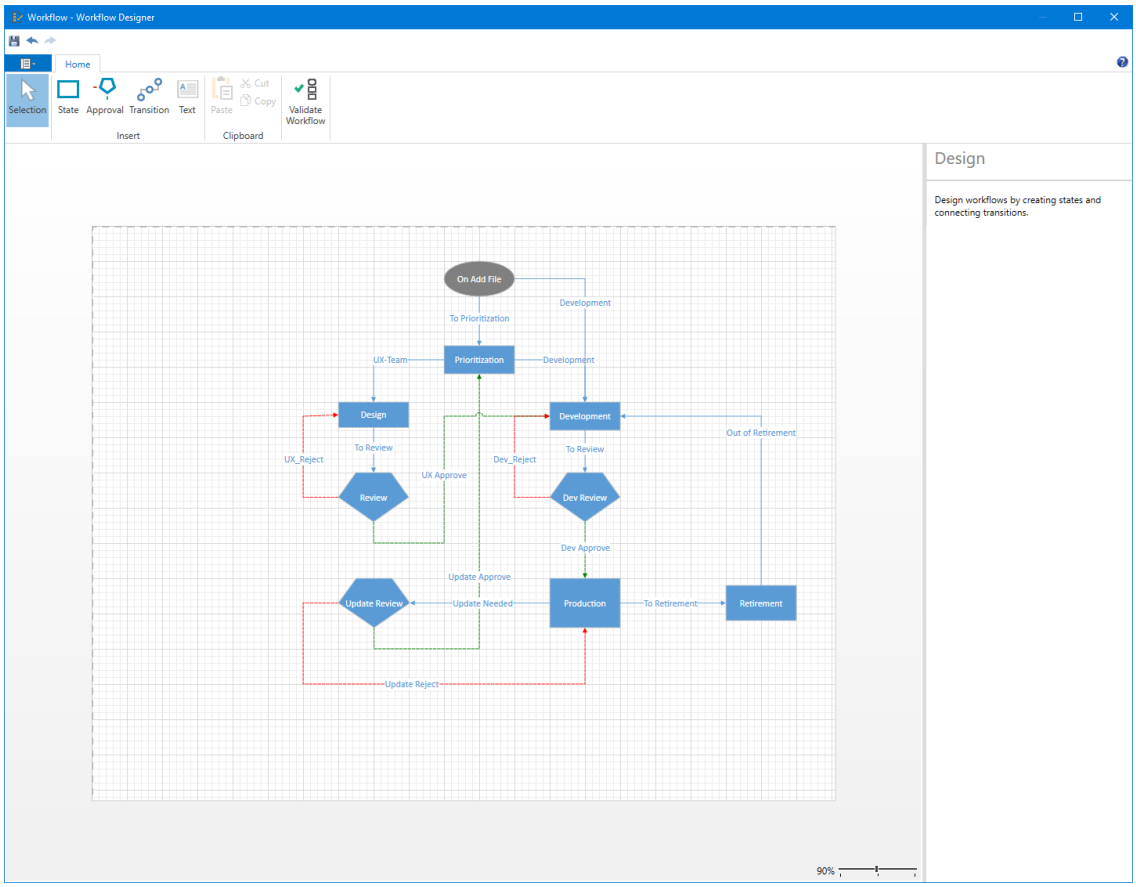

Figure 7. View of the workflow designer.

3.4 Insomnia

For creating and testing the REST API calls, an open-source API development platform called Insomnia was used. It is broadly used by multiple companies, including Netflix, Sonos & Tesla, for building, designing, and testing APIs. Key features of the Insomnia are multiprotocol support without app switching (HTTP, REST, GraphQL, SOAP etc.), developer friendly UI, collaboration with other developers by using local, cloud or Git storage and automation which supports automatic linting and creating custom API tests that can be used to test and validate functionality of built APIs. [14].

In the project test environment, Insomnia was used to build, send, and test the REST API requests. Building included creating data contents like the method used, endpoint URL and the request body itself. Sending the request with Insomnia would show the status and response of the sent message. These results will be discussed in more detail later in this chapter.

3.5 REST API Printing

Bartender has multiple ways of implementing the usage of REST API. In this section, three of those methods are introduced and analysed in order of their discovery.

3.5.1 Revision 1

The First revision of the REST API integration was done with the Bartender Integration Builder introduced in section 2.5.3. It utilized a web service integration that monitored for a web service request to arrive and then ran through predefined actions from Bartender Actions API. In this piloting phase, the only interest was in label printing, so the only action used was the Print Document action. Building this service was easy as multiple step-by-step instructions were found on the topic. Below, in the Figure 8 UI of the Integration Builder can be seen.

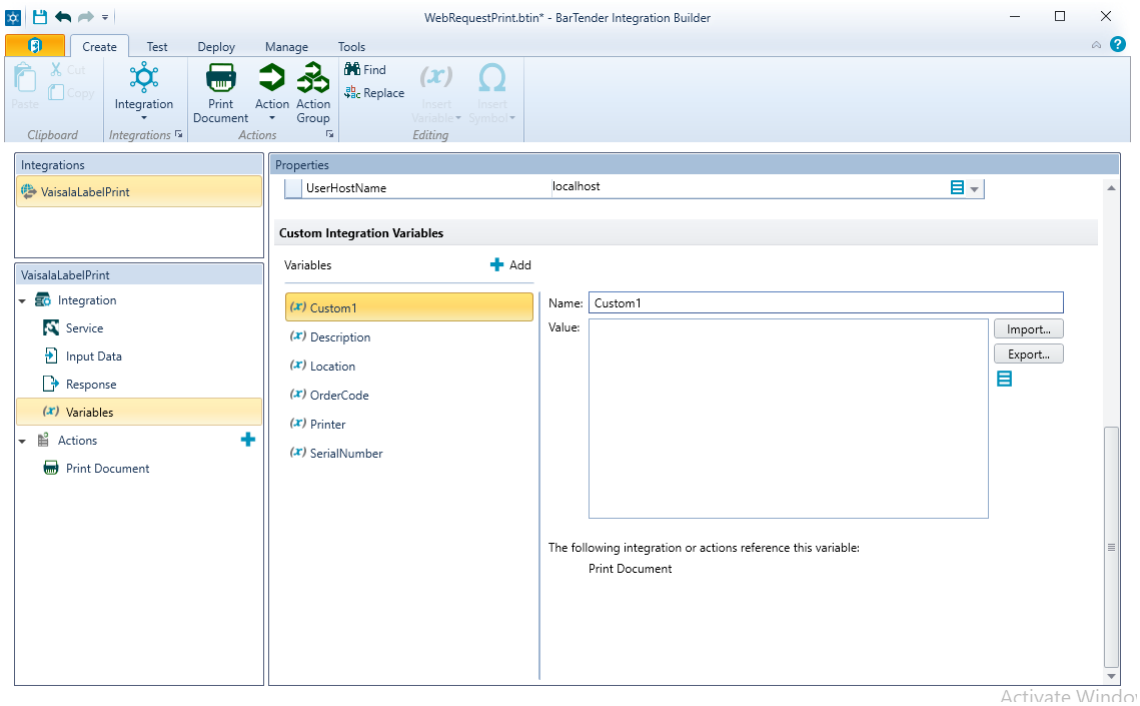

Figure 8. Integration Builder, Variables tab UI view.

In the Service tab, name of the web service and used port is configured. Based on these setting, the URL of the service is created. In this testing case, service name was “LabelPrint”, and port used was the default and suggested 80 creating URL of *http://<host>:80/Integration/LabelPrint/Execute*. Host would naturally be the server’s name where the integration is deployed. Different integrations would use the same URL, only separated by the service name.

In the Input Data tab, the format of the incoming data needs to be defined and in this case, format chosen was JSON. In the Response tab, the format and the content of the service request response is defined. Once again, JSON format was selected and “action summary” as the contents of the response. Action summary is an automatically generated data response from the web service. An example of the response can be seen later in this chapter.

In the Variables tab, all the used variables need to be introduced. All variables used in can be seen in the Figure 8 above, and they are listed and described in the Table 3 below. The only mandatory variable is the Location, without a file path no label can be printed. Printer variable can be considered as a semi-mandatory field, as if left empty, the default printer saved to the label template is used. The

problem is that multiple stations could be using the same template, meaning that the label would possibly be printed to the wrong location.

Table 3. Used parameters in the first revision REST API setup.

Variable	**Description**
Location	File path of the label template.
Printer	Used label printer name.
Custom1	Custom data field.
Description	Product name, type, etc.
OrderCode	Configuration code of the product.
SerialNumber	Serial number of the product.

The only action used in this simple web service was Print Document. Within the action settings, usage of named data sources must be selected and specified as seen in the Figure 9 below. The percentile symbol around the parameter name tells the integration that it is an empty variable which will be populated via the web request message.

The location of the document and printer used was also configured in the document tab seen hidden in the Figure 9 by using %Location% and %Printer% parameters. As with named data sources, they are treated as empty variables, which are then populated by the request message. This makes it possible to print any document whether it is in the Librarian, local or virtual drive and use any printer installed on the source PC.

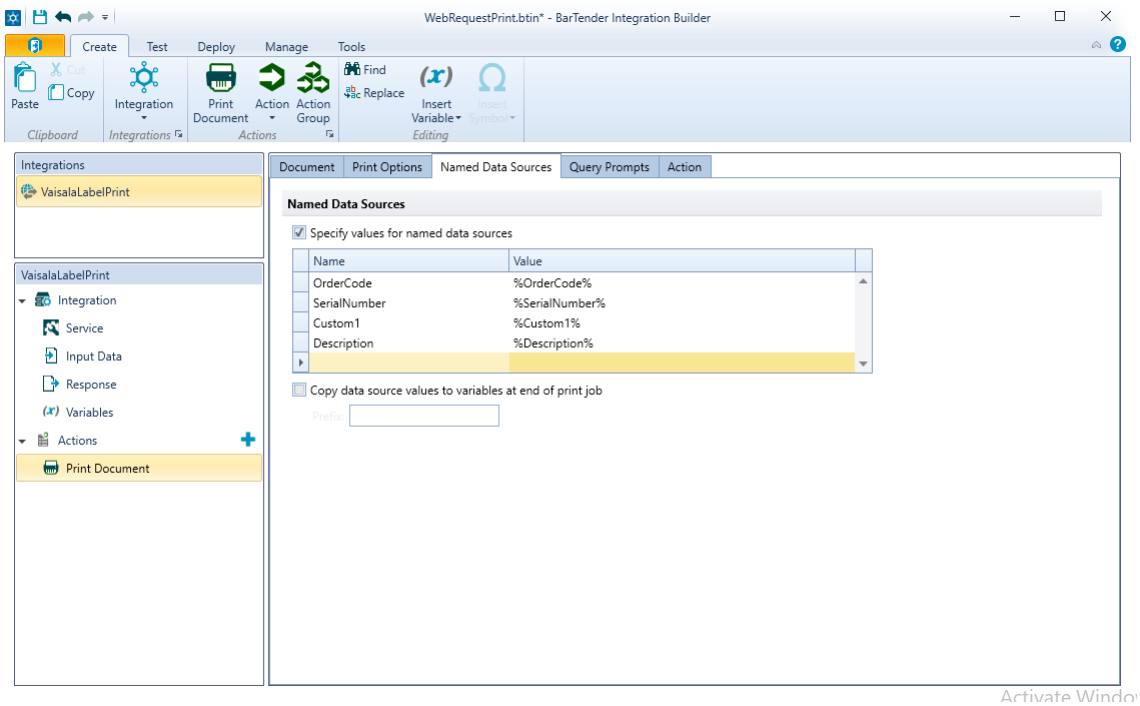

Figure 9. Integration Builder, Named Data Sources UI view.

Below, in the Figure 10, results of the test printing request can be seen. In the middle, contents of the sent JSON request is visible, and on the right is the response from the host server. In the JSON request, the location of the label template and the named data sources and their data are sent to the host server. Based on this JSON query, the label template is selected, and its predefined data slots are populated and then printed with the printer defined in the template settings.

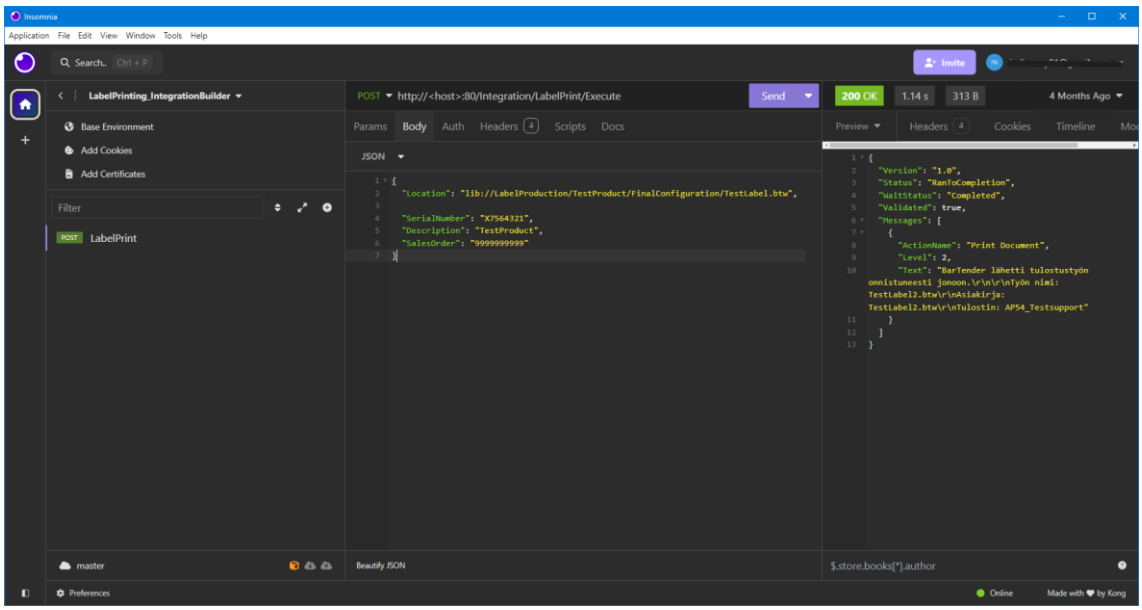

Figure 10. Printing request and response via Insomnia.

3.5.2 Revision 2

The second revision of the REST API was done with the Bartender REST API introduced in chapter 2.5.1. It resembles the first iteration a lot, which is not a surprise as the same actions are used and are available as with the Bartender Integration Service. The main difference is that a single endpoint with URL *http://<host>:5159/api/actions/* is capable of running all the different actions, instead of having to create multiple integrations. Technical difference here is that the POST request itself contains the actions used and their parameters.

An example JSON printing query can be seen in the Figure 11, where the middle part is the sent request body. It consists of the action used (line 2, PrintBTWAction) and below it, in curly brackets, all the parameters. From line 4 to line 9, the basic parameters are defined. Printer defines the printer used in the printing job. This field is not mandatory but recommended, as the printer saved in the label template might not be the same as the desired printer. DocumentFile specifies the .btw document location, which can be anything from Librarian library directory, to a local file path. DocumentFile parameter is mandatory for a printing job. Copies parameter should be self-explanatory, and PrintJobTimeout defines the request timeout in milliseconds.

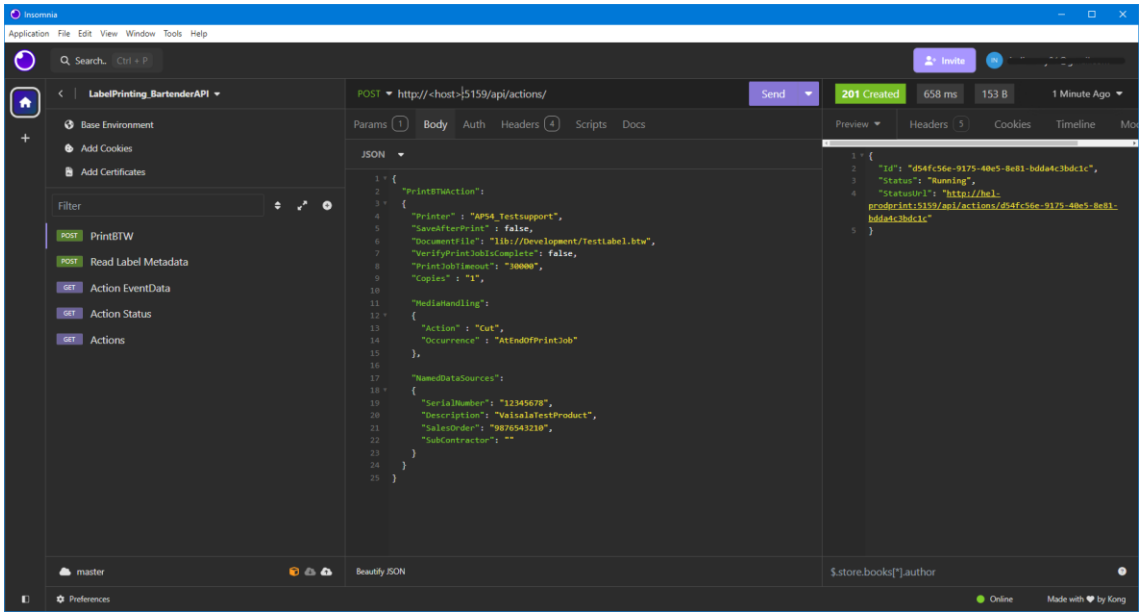

Figure 11. Print request and response using Bartender REST API via Insomnia.

From line 11 forwards, some optional settings are defined. With MediaHandling command, post printing actions can be controlled. In the example seen in the Figure 11, the label is cut after every print job, meaning for example if five labels are to be printed, all of them would be cut separately. Multiple MediaHandling actions are available depending on the hardware of the label printer. After line 17, the NamedDataSources are populated. This is also optional but in 99% of the cases a used feature. The name of the parameter, for example SerialNumber needs to match the named data source defined in the label template. If this data source is populated or not, does not matter. In a case of an empty string, the label data field would be left blank. In case data field is defined in the label template, but not populated in the request body, the default value saved in the label template is used, which normally would be an empty string or integer.

Response of the POST request can be seen on the right side of the Figure 11. This response only tells the system if the creation of the request was successful or not. It does not contain any information about the request action itself. By following the StatusUrl parameter link, the status of the action itself can be monitored. Example contents of this status message can be seen in the Figure 12 below.

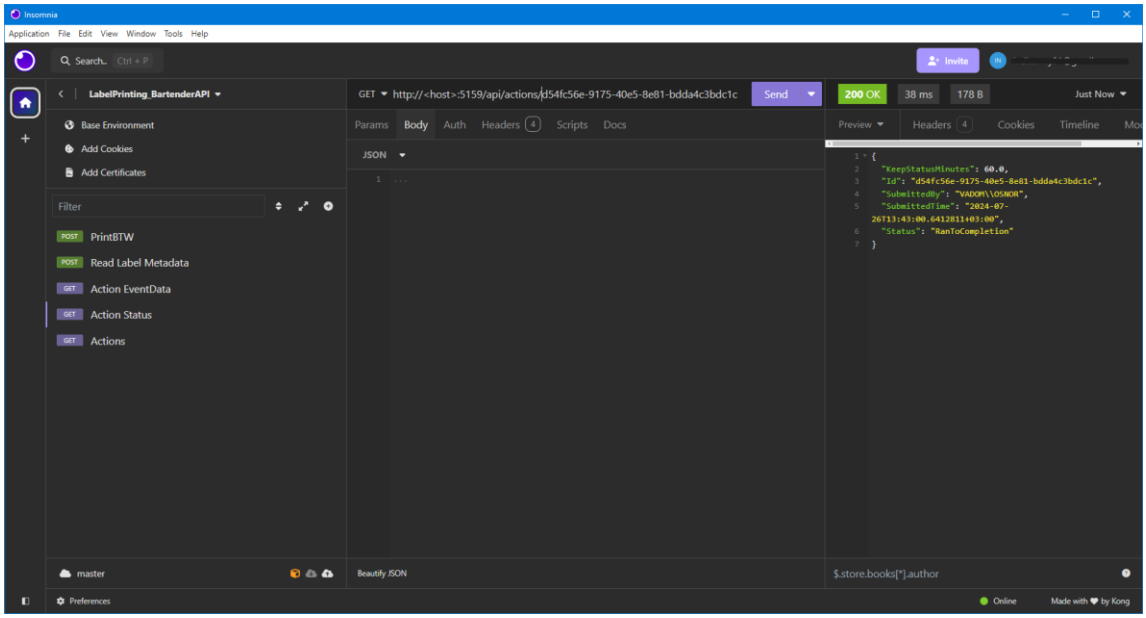

Figure 12. Response of status query of a single action.

More detailed information of an posted action can be queried with GET method of *http://<host>:5159/api/actions/*. As it can be seen, the URL is the same as with the POST method. The returned information can be delimited by adding a parameter name containing the data at the end of the URL, for example *http://<host>:5159/api/actions/EventData*. Having to query the actions separately is not optimal, and this factor is the definite drawback of using this REST API design.

As a bonus, ReadFileAction request was created, which allowed for reading the metadata of a chosen label template. Contents of the request body can be seen in the Figure 13, and the action response message in the Figure 14.

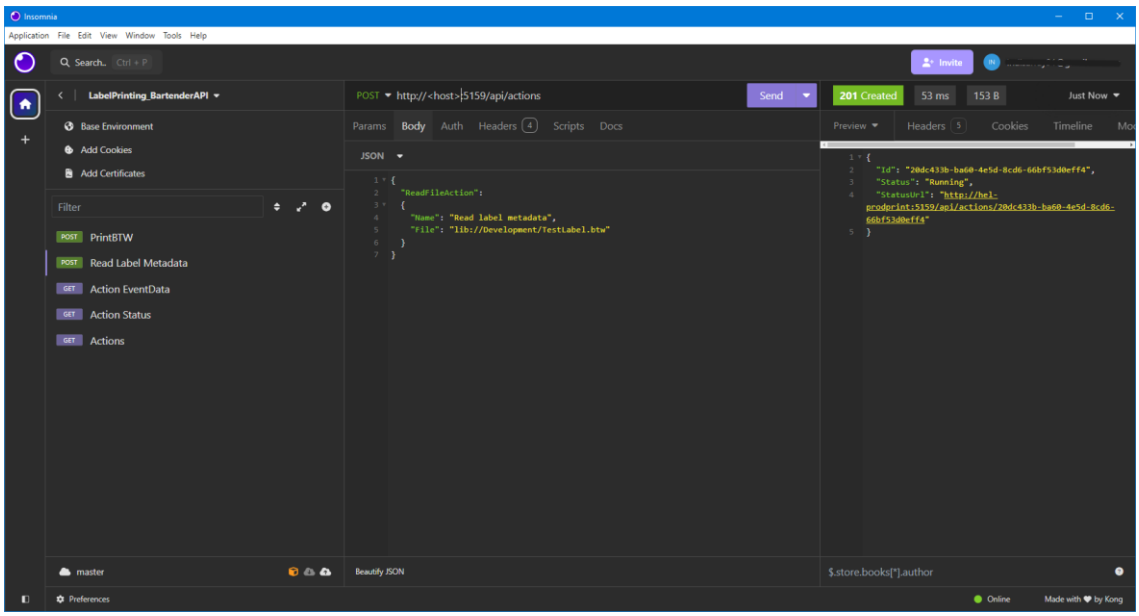

Figure 13. Query for reading label template metadata.

In the end, there was not any reasonable use for this metadata query method, as most of it was unreadable, so it was noted as useless. However, it shows that different kinds of methods can be created to control or query all sorts of data.

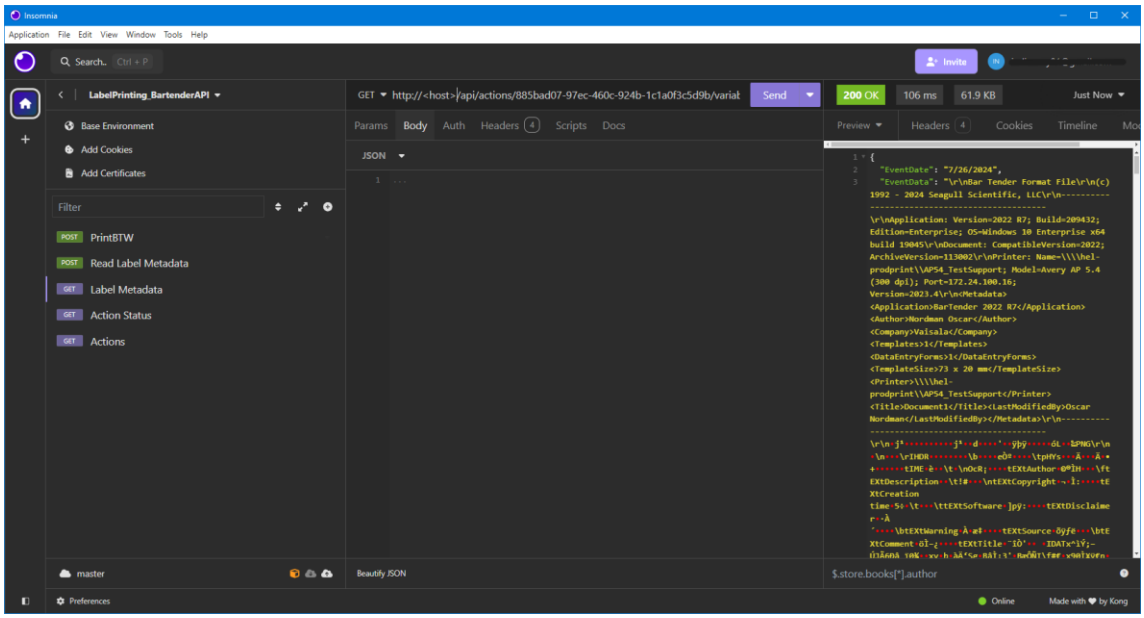

Figure 14. Response of the label metadata query.

3.5.3 Revision 3

The third and final revision was based on Print Portal REST API introduced in chapter 2.5.2. It is clearly the simplest approach to the REST API matter. It consists of four endpoints with different actions which were listed in Table 2. No customization is available, no need to predefine used actions or having to post complex queries. Architecture wise, this third version is the truest to the set of rules usually linked to REST API design, briefly discussed in chapter 2.5.

Basic idea behind this design is that the Print Portal consists of multiple libraries with their own static valued IDs. All the available libraries can be queried with a GET method *http://<host>/Bartender/api/v1/libraries/* seen used in the Figure 15 right below.

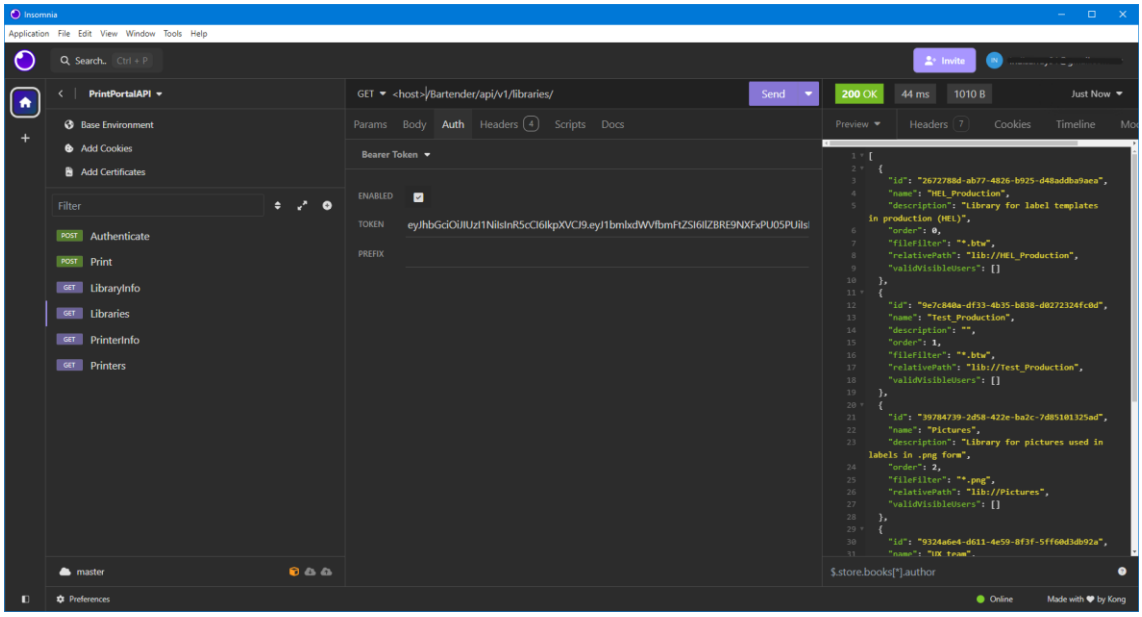

Figure 15. Query of all the libraries in use.

By adding the chosen library ID at the end of the URL, its content can be queried like seen in the Figure 16. This structure makes managing the label templates quite simple, as all the labels would be clearly separated to different libraries based on their status or intended use, for example production, development, pictures, etc. The production library would naturally consist of labels that are in production use, the development library could be used as a test field for new or

upgradeable labels and finally the pictures library would consist of different images that are used populating label fields consisting of an image like for example a subcontractor logo.

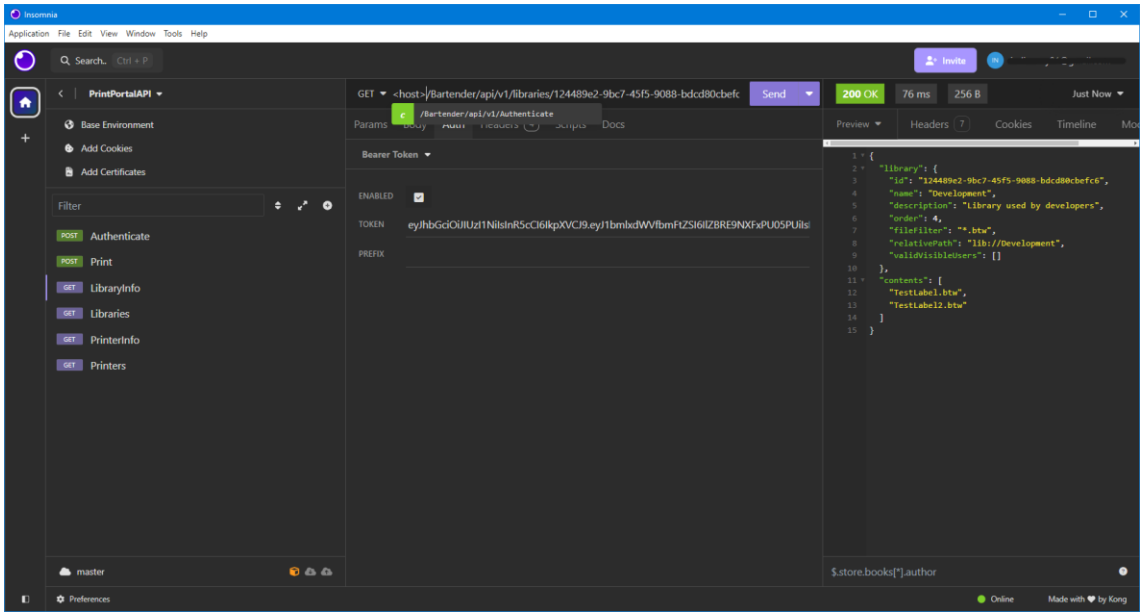

Figure 16. Content query of the production library.

With Print Portal REST API, use of authentication is also available. By using POST method *http://<host>/Bartender/api/v1/Authenticate*, an authentication token can be queried, which generates a bearer token as a response. This authorization token is then needed in the future endpoint queries as a header, otherwise the system server will return an authorization error. An example authentication token query can be seen in the Figure 17 below. The request body itself only contains two parameters, which are the login credentials username and password. Access permission of the given credentials to Print Portal are checked from the main system server. If the credentials have the access, an authorization token would be returned, which is valid for a certain period configured in the system settings. In case of no permission, the response will inform the user of inadequate access rights.

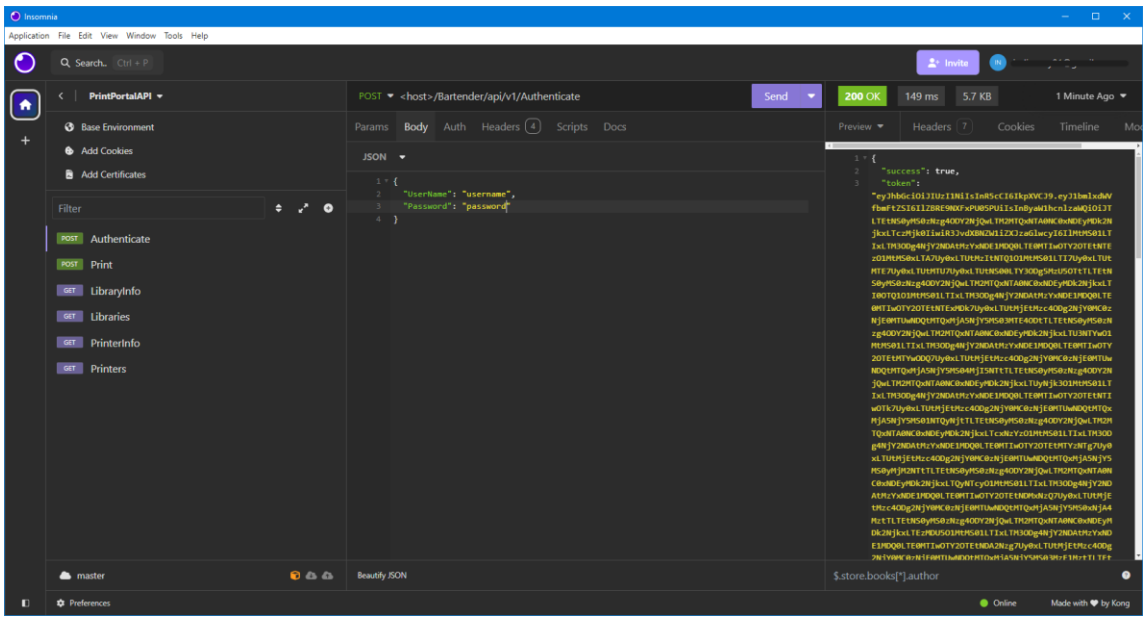

Figure 17. Authentication token query and response.

The last endpoint is the printing command, which uses a POST method and a URL of *http:///<host>/Bartender/api/v1/print*. Once again, the query contains the basic parameters like used printer name, the number of copies to print, and the named data sources used to populate the label template data fields. The biggest difference within the JSON body is, that a relative path of the chosen document can be used, instead of an absolute path used with other Bartender REST API options. Example of this can be seen in the Figure 18 below. This makes the file paths of the label documents a bit simpler, and the overall directory logic can also be simplified.

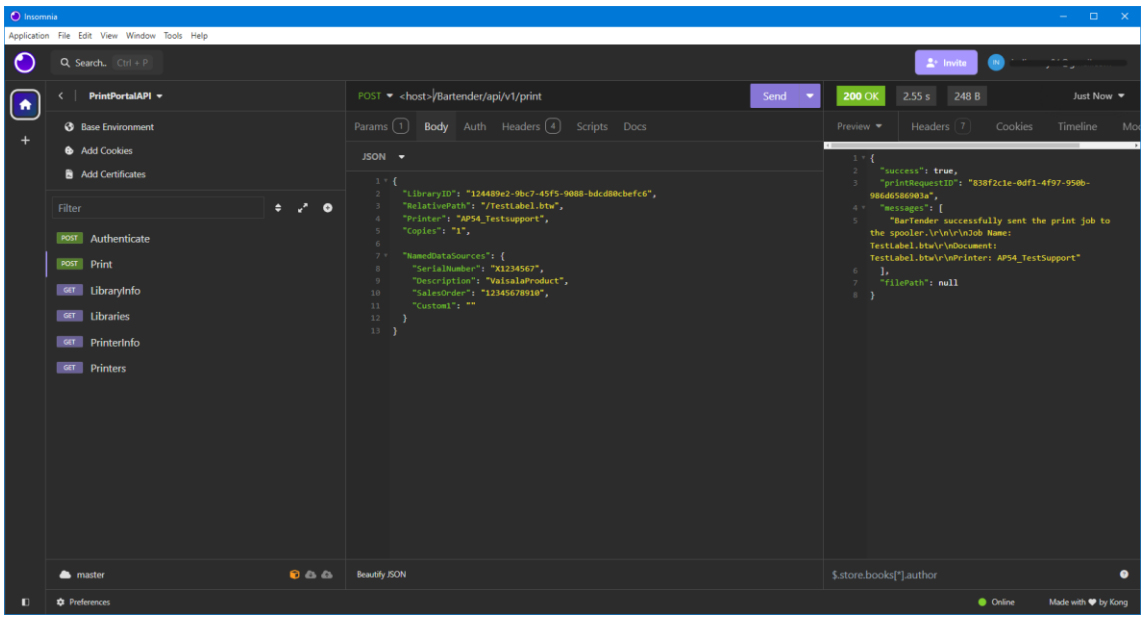

Figure 18. Print Portal REST API print query and response.

A big advantage compared to the Bartender REST API is, that the response messages are much more dynamic and informative. In POST method cases, the responses don not just contain whether the creation of the query was a success or not like in Bartender REST API, but also updates the status of the query automatically. This means that no separate query is needed to get the printing job status after sending the printing command. Also, based on the initial testing, it seems that the Print Portal REST API is capable of giving more detailed status responses in case of a system error.

4 Results

This chapter focuses mainly on the results that were discovered during the piloting phase of the Printing Service project. The implementation phase was a bit delayed due to the massive amount of investigation work needed to get the whole system up and running as specified.

4.1 Results of Piloting

Piloting and testing the new Bartender 2022 Enterprise system with REST API resulted in three different implementation options. All the three options were

introduced to a small group of test engineers and architects. Below are some tables to summarize the advantages and disadvantages of each option.

Table 4. Advantages and disadvantages of using Bartender Integration Service.

Advantages	**Disadvantages**
Easy to setup and manage	Dependant of the Integration Builder
Integration easy to relocate	All actions need their own integration
Action customization options	All parameters need to be predefined
Tools for testing created services included in the Integration Builder	Full library file path needed for printing

My first iteration of the REST API design was created with the Bartender Integration Service. It was an easy and simple process, including useful testing capabilities. The biggest drawbacks using Integration Service was, that all created methods all run as separate services and that all the used parameters needed to be predefined. Considering the variety of products Vaisala produces, and the amount of different manufacturing processes used, made the latter disadvantage a difficult obstacle to overcome. Thus, another more dynamic version of the REST API was needed.

Table 5. Advantages and disadvantages of using Bartender REST API.

Advantages	**Disadvantages**
Single endpoint is capable of different actions	JSON request body is quite complex
Used parameters don’t need to be predefined	Action status needs to be separately queried
Post printing actions can be affected (label cutting, etc.)	Full library file path needed for printing
Action customization options	

The second iteration of the REST API design was based on Bartender REST API. The usable actions remained the same as with Integration Service, but the whole RESTful concept was different, with multiple advantages over Integration Builder. Firstly, a single web service could handle multiple different actions via a single endpoint instead of having to run multiple services for different actions. Secondly, predefinition of used parameters was not needed, as many have either default values or are just not used if left empty. Also, a nice touch was that post printing actions could be affected by the query, meaning for example label cutting settings could be altered on the go.

The single endpoint design comes with the disadvantage of making the request body fairly complex. It needs to include the action name used, its parameters and possibly other data depending on which action is in question. This also meant that the management of the query content was not that simple and might have needed some logic behind it. I have also listed the need to query the POST method status separately as a drawback, even if this is quite common in REST APIs. This is because the other revisions can send the action status automatically to the user, and not just the response whether the request was successfully created or not.

Table 6. Advantages and disadvantages of using Print Portal REST API.

Advantages	**Disadvantages**
Simple operation	Limited actions available
More secure	No customization
Enables simplified library structure	
No predefinitions	
Automatic action status response	

The third and the final iteration was based on the Print Portal REST API. The clear advantage over the other two versions is its simplicity. Four simple commands are available, of which only one uses a POST method. Request body only contains data of the used printer, relative file path and the named data

sources. The Print Portals libraries and document metadata also allows for more simple directory structure, and it is a bit more secure as authentication can be enabled, making it only possible to use the given REST methods with a security token as a header. The biggest disadvantage compared to other revisions is the limited methods available and customization options. However, it has all the needed functions and more might be released in the future versions by Bartender.

Based on the testing of the three REST API options and the gathered information, Print Portal REST API was selected as the base of the refinement work to create the final integration. It was simple and fast due to being optimized to the Print Portal service, which will be a significant tool for most of the people working at Vaisala. At the end of the testing phase, it was also discovered that using Print Portal based REST API allows for even more simple library structures, as Print Portal can read description data that has been saved to the label template. For example, products using a specific label template can be listed within the template description data and that can be used as a filter in the Print Portal system, as can be seen in the Figure 19 below.

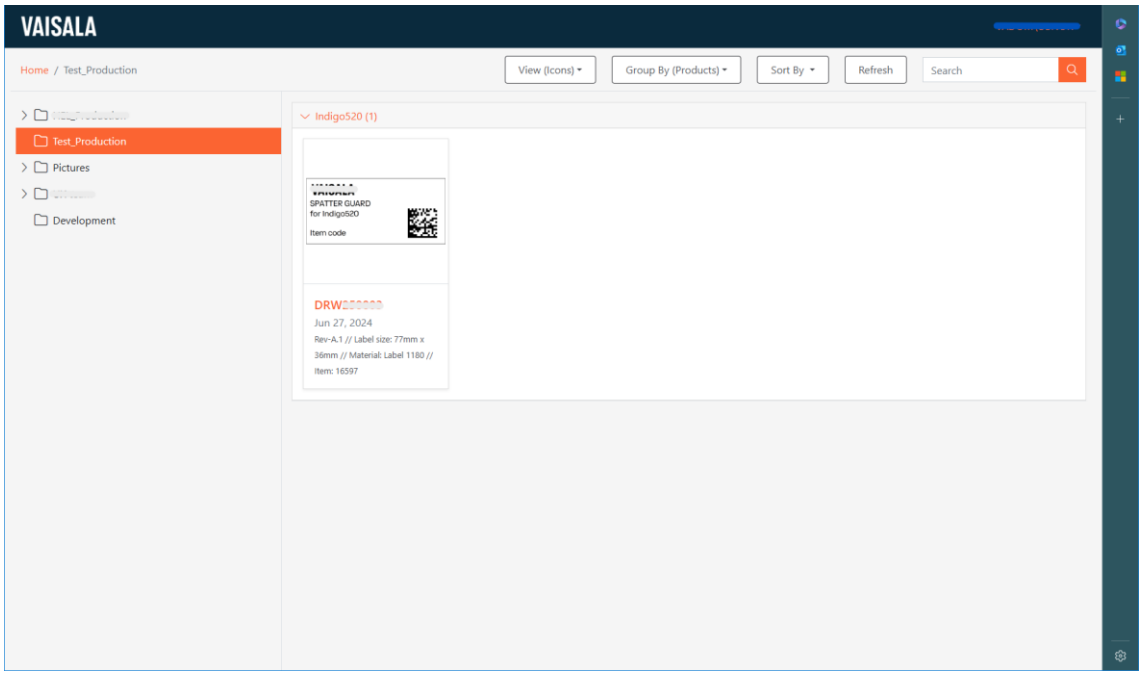

Figure 19. View from Print Portal with description data in use.

4.2 Implementation Process

Once the first stage of the testing was concluded and the Print Portal REST API was chosen as the base for the Printing Service, the actual development started. This included implementing the REST methods to the Vaisala platform, building the whole system from ground up to the production network and conducting second stage testing. Implementation was done by in-house platform development team. Setting the Bartender system and the database up, and conducting the final testing was my responsibility. The most challenging part of this “roll-out ready” stage, was to build the final libraries and their directories and to standardize and simplify as much as was possible. The creation work of the final label printing system was not difficult but coming up with logical solutions that would work with multiple different processes and products was the hard part. In the end, with help of senior test engineers and architect’s pretty optimal solutions were found. Testing the final system was done with a dummy test software created by me, using the implemented REST methods in Vaisala’s platform. Testing went well, no major problems occurred, although some potential improvements were already found, concerning mostly the logging of the whole system and reprinting a failed label. More of this is explained in the next chapter. As for now, the Printing Service is roll-out ready and first roll-out projects have been started. Unfortunately, no feedback or results did make it in time to be discussed in this thesis work.

4.3 Improvements in the Future

As mentioned in the previous chapter, during the test phase, it was found that the logging information is not quite optimal. The way the data is saved means handling the logged data is a bit difficult. For example, extracting single printer’s log data based on time period of e.g. 30 days cannot be done, which is basically the most important data the management level would like to have. This is because the host server is always shown as the requester of a single printing job and not the production unit that truly sends the printing request. Most probably this has something to do with the way the label printers are set up.

In the future, when the Printing Service utilization rate will be much higher, some slowness and unresponsiveness might occur due to the high level of requests coming to the system server. However, this is already an identified problem, and the solution is to optimize the server and service settings, and to add more workers to the system database, meaning that the database will be able to process as many queries at the same time as is the worker limit.

5 Conclusions

The goal of this project was to implement REST API architecture for product label printing and to get it roll-out ready for production use. This goal was completely achieved, and the first Printing Service roll-out projects started as this thesis work came to an end. Additionally to this thesis project, I also created internal documentation and did few presentations of the topic to raise awareness of the upcoming new feature. Initial receptions were mostly positive, and I received commendation of the created documentation of the whole system.

By the time I started with the project, I had basically zero knowledge of REST APIs. The used Bartender environment was already familiar to me, which helped in the configuration process and generally understanding the logic behind the whole system.

References

1 Bartender 2016 EOL. Online Source. Seagullscientific. <https://www.seagullscientific.com/support/2016-end-of-support/>. Accessed June 17, 2024

2 Vaisala internal project description link. Online Source. Vaisala. <https://confluence.vaisala.com/confluence/display/PT/Product+labeling+and+printing+Service+Description+Document>. Accessed April 25, 2024

3 Bartender 2021 Product Reference Guide. Documentation. Seagullscientific. <https://www.seagullscientific.com/media/3058/bartender-2021-product-reference-guide-en-prt-0072_1021.pdf>. Accesses April 4, 2024

4 Helper page for Bartender Administration Console. Online Source. Seagullscientific. <https://help.seagullscientific.com/2021/en/Subsystems/AdminConsole/Content/Admin_Console_Main.html>. Accessed April 14, 2024

5 Bartender Librarian, Technical Documentation. Documentation. Seagullscientific. <https://www.seagullscientific.com/media/4145/bartender-librarian_2021.pdf>. Accessed April 15, 2024

6 Bartender Print Portal, Technical Documentation. Documentation. Seagullscientific. <https://www.seagullscientific.com/media/3214/bartender-print-portal.pdf>. Accessed April 17, 2024

7 What is a REST API? Online Source. IBM. <https://www.ibm.com/topics/rest-apis>. Accessed May 12, 2024

8 Bartender REST API Overview. Online Source. Seagullscientific. <https://support.seagullscientific.com/hc/en-us/articles/4413434027799-BarTender-REST-API-Overview>. Accessed May 19, 2024

9 Print Portal REST API. Online Source. Seagullscientific. <https://support.seagullscientific.com/hc/en-us/articles/360053982393-Print-Portal-REST-API>. Accessed May 20, 2024

10 Integration Builder Overview. Online Source. Seagullscientific. <https://help.seagullscientific.com/2021/en/Subsystems/IntegrationBuilder/Content/IntegrationBuilder_Main.html>. Accessed April 22, 2024

11 Bartender License Management. Webinar. Seagullscientific. <https://support.seagullscientific.com/hc/en-us/articles/360022950393-License-Management-in-BarTender-2019-and-Later>. Accessed April 25, 2024

12 Understanding the Bartender System Database. Documentation. Seagullscientific. <https://www.seagullscientific.com/media/1368/bartender-system-database.pdf>. Accessed April 26, 2024

13 Bartender Librarian. Documentation. Seagullscientific. <https://www.seagullscientific.com/media/1351/bartender-librarian.pdf>. Accessed May 13, 2024

14 Insomnia homepage. Online Source. Insomnia. <https://insomnia.rest/>. Accessed April 29, 2024