Developing a connector for the Madbot API

Jaffar Gura




Université Paris Saclay
Master of Bioinformatics and Biostatistics

Developing a Connector for the Madbot API

Presented by

GURA Jaffar

Internship supervisors: Imane Messak *Institut Français de Bioinformatique*
Baptiste Rousseau *Institut Français de Bioinformatique*

June 26, 2024

Contents

1 Introduction

1.1 Research Institute

The internship was conducted within the **IFB-Core** team (UMS 3601, CNRS) at the **IFB** (*Institut Français de Bioinformatique*). I was welcomed by the **Madbot** (*Metadata And Data Brokering Online Tool*) team. IFB is renowned for its contributions to the field of bioinformatics, providing critical infrastructure and tools for data management and analysis.The main missions of the IFB are:

- **Deploy Physical and Software Infrastructure**: Establish and maintain physical and software infrastructure for bioinformatics services.
- **Support Research Programs**: Provide support for research programs in biology, health, environment, biotechnology, and agronomy through shared expertise and skills.
- **Offer Continuing Education**: Provide ongoing bioinformatics training for biologists and bioinformaticians.
- **Develop Strategic Vision**: Maintain France at the highest level of expertise in biological data analysis and provide access to cutting-edge bioinformatics technologies.
- **Innovate Tools**: Develop innovative tools to address the challenges of integrative bioinformatics.
- **Facilitate Ambitious Research Projects**: Serve as a lever for the conception and implementation of ambitious national research projects.
- **International Representation**: Represent the French bioinformatics community internationally, particularly within the European network **ELIXIR** (*European Life-Science Infrastructure for Biological Information*) [ELIXIR, 2024].

The team comprises experts in bioinformatics, data science, and software development, working collaboratively in an agile framework(Scrum) to enhance data accessibility and usability for researchers.

Scrum is a specific Agile methodology that focuses on iterative development, flexibility, and close collaboration with a working group. Projects are divided into small units called sprints, typically lasting two to four weeks. This approach allows teams to frequently reassess and adapt their plans, ensuring continuous improvement and responsiveness to change, ultimately enhancing the quality and usability of their outputs [Beck et al., 2001].

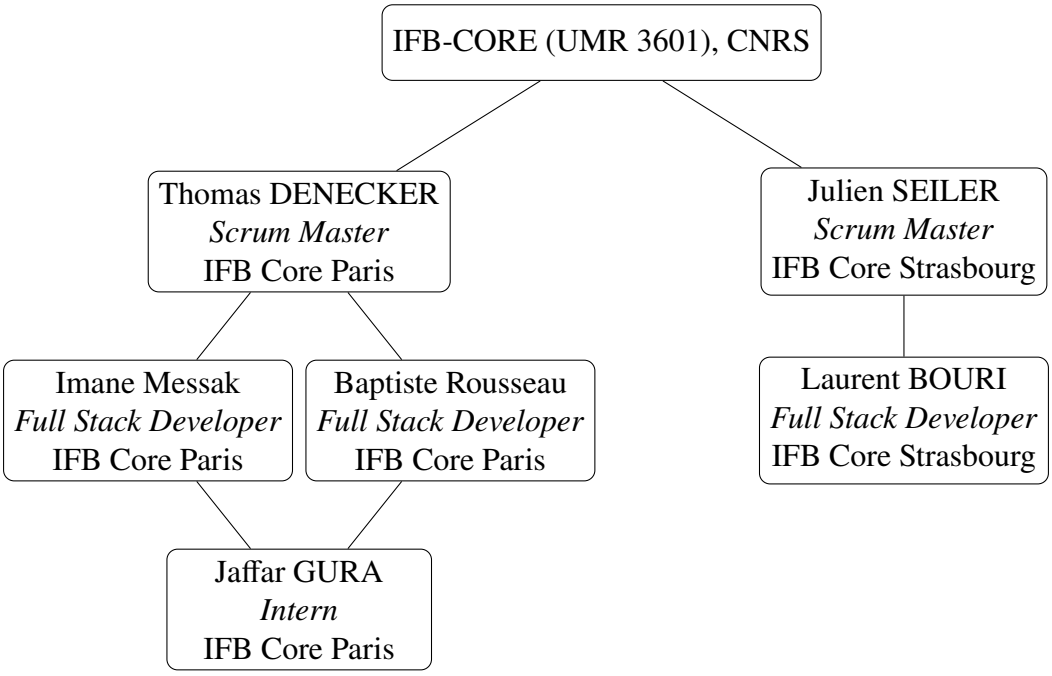

Figure 1: Organizational Chart of the Madbot team

A Scrum team typically consists of a Product Owner, Scrum Master, and Development Team. The Scrum Master facilitates the Scrum process, ensuring that the team adheres to Scrum principles and practices. The Development Team is a self-organizing group of professionals who deliver the product in increments.A simplified organization structure of the team is shown in Figure 1.

1.2 Problem in Bioinformatics Data Management

1.2.1 Data Diversity and Fragmentation

Managing and organizing large amounts of data from different sources is a significant challenge in bioinformatics. This is because there are many types of data, like DNA sequences, protein structures, gene expression profiles, and clinical data. Each type needs specific tools and formats for storage and analysis, which makes data integration difficult. Scientists use different systems to store and analyze their data, leading to inefficiencies. They spend a lot of time ensuring different formats and tools are compatible. The lack of standard protocols makes it even harder to keep data consistent and reliable.

1.2.2 Data Volume, Integration, and Interoperability

High-throughput sequencing technologies produce huge amounts of data, making real-time or near-real-time data management challenging. Traditional data management systems often can't

handle this large scale efficiently. Integrating data from various sources, such as public databases, laboratory management systems, and electronic lab notebooks, is crucial but complex. These sources have different access policies, formats, and update schedules, making integration time-consuming and error-prone. Tools used in bioinformatics need to work together smoothly for effective data exchange and analysis. However, many tools are developed independently, leading to compatibility issues that slow down research progress and make collaboration harder.

1.3 Madbot's Approach

Madbot is an open-source project available on GitLab,that offers a dashboard designed to manage research data and metadata. The application aggregates metadata related to scientific projects and accurately identifies the data's storage locations via multiple connectors. Its primary goal is to assist researchers in publishing their data and metadata to numerous scientific repositories.

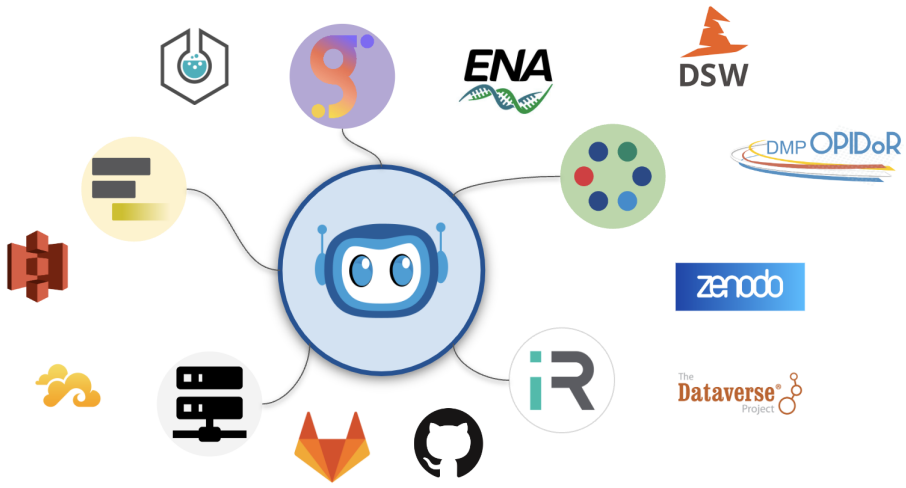

Figure 2: Various bioinformatics tools with which Madbot is connected and will connect in the near future. The currently accessible connectors are SSHFS and Galaxy. Labguru and Omero are under transformation, while iRODS is still a work in progress.

Madbot connects to several key platforms, including:

- **SSHFS**: Connection to Clusters and NAS (Network Attached Storage) Servers.
- **Galaxy**: An open-source platform used for data-intensive biomedical research [The Galaxy Project Team, 2024].
- **Labguru**: An electronic lab notebook that helps scientists manage their lab work, inventory, and data [BioData Inc., 2024] (currently under transformation).
- **ENA** ***(European Nucleotide Archive)***: A data warehouse that stores large amounts of nucleotide sequencing data [Yuan et al., 2023].

- **Zenodo**: A research data repository that allows scientists to share and preserve their research outputs, including datasets, software, reports, and more [Zenodo, 2024].
- **Omero**: A platform for visualizing, managing, and analyzing large sets of image data [Allan et al., 2012](currently under transformation).
- **iRODS**: A data management software that enables users to handle large-scale data [Rajasekar et al., 2010] (work in progress).

The currently connected and those that are under development are illustrated in figure 2 While these platforms excel individually, they often operate in isolation. Madbot integrates these tools, allowing scientists to access and manage their data from a single interface. This integration streamlines data management and submission, enabling researchers to focus on their scientific work rather than dealing with fragmented data sources.

1.4 Standardization

Madbot adheres to the **ISA** *(Investigation, Study, Assay)* standard, ensuring that data and metadata are organized consistently. This adherence promotes good data management practices and helps achieve **FAIR** [Deutz et al., 2020] *(**F**indable, **A**ccessible, **I**nteroperable, and **R**eusable)* publication standards. The ISA model is developed and supported by ELIXIR, an intergovernmental organization aiming to establish and maintain a sustainable infrastructure for biological information in Europe. Since IFB is the French node of ELIXIR [ELIXIR, 2024] and ENA is one of the core nodes of ELIXIR, it is natural for IFB to use the ISA model and for ENA to seek its implementation.

1.5 Objectives of the Internship

The primary objectives of the internship were to:

- Develop a unified connector module that can handle various protocols.
- Simplify the management of connections to different data sources.
- Ensure the new module provides a consistent interface for the Madbot front end.
- Improve the performance and scalability of data interactions in Madbot.
- Adhere to data management standards to ensure high-quality and reliable data integration.

2 Materials and Methods

2.1 Development Tools and Environment

Our team is developing the project using Python 3.10 because of its readability and versatility. We are using Visual Studio Code (VSCode) as our code editor and managing software dependencies with Conda to keep a consistent development setup. The **API** (*Application Programming Interfaces*) is built on Django 5.0.3, a popular web development framework, along with **DRF** (*Django Rest Framework*) 3.14.0 to handle the API endpoints [Django Software Foundation, 2023]. APIs allow different software systems to communicate with each other, improving the application's interoperability. On the client side, the web application is built on Vue.js, a progressive JavaScript framework.

For the database management system, we are using PostgreSQL 16.0. Django's **ORM** (*Object-Relational Mapper*) is making it easy to interact with the database using Python code. We are doing all the development on MacOS Sonoma 14.5 and deploying the application on Ubuntu 20.04 **LTS** (*Long Term Support*).

We are managing project configurations and dependencies using a **TOML** (*Tom's Obvious, Minimal Language*) file, which is simple and human-readable, ensuring consistency across different development environments. We are using Git for version control, with our repository hosted on GitLab. This setup is facilitating efficient collaboration and tracking of code changes. You can access the repository at *https://gitlab.com/ifb-elixirfr/madbot*.

2.2 Building the Connector Module

The connector module, designed to link the main application with external data sources, was built using the `fsspec` library (version 2024.6.0) [Durant, 2024]. This library provides a unified interface for interacting with various file systems, whether local or remote. Additionally, `sshfs` and `asyncssh` (both version 2024.6.0) were used to establish secure connections to remote servers and NAS, and to execute custom scripts on remote servers, thereby automating various tasks.

We began our development process by creating a clone of the SSH connector that was previously available, without changing the underlying architecture, as a proof of concept. The original SSH connector was designed to interact with remote file systems via the SSH protocol. This initial step was crucial as it allowed us to test the feasibility of using `fsspec` for our unified connector. By cloning the existing connector, we established a baseline for performance and functionality, ensuring that `fsspec` could handle the necessary SSH interactions. This approach minimized the risk associated with major architectural changes, allowing us to incrementally adapt the existing, proven system and maintain core functionalities during the transition.

3 Results

3.1 Prototyping Stage

During the dry running the SSH Connector clone, I encountered several issues:

3.1.1 Permission Issues and Solution Implementation

During the implementation, I discovered that, similar to the `paramiko` implementation, this solution allowed visibility of objects that had POSIX permissions but not ACL (Access Control List) permissions. Consequently, files, folders, and links appeared in the directory listings even though they did not have the necessary permissions to access them. This resulted in a cluttered user interface, displaying items that users could not interact with, and potentially posed a security risk in other use cases.

To address these issues, I implemented an additional verification step using asynchronous SSH (`asyncssh`). This solution involved running a bash script on the cluster in parallel with the `fsspec` requests. The script performed the following functions:

- **Permission Verification:** It checked the file permissions of each item, ensuring that only files, directories, and links that the user is permitted to see were included in the results.
- **Enhanced Data Retrieval:** For cases where the `fsspec` data object returned `None` for MIME types, the script also retrieved the MIME types of the files.

The use of `asyncssh` allowed these checks to be executed in parallel, minimizing performance overhead and ensuring that the directory listings returned by our application were both accurate and secure. Additionally, the script provided a way to piggy-back more commands in case there is need for other fields not provided natively by `fsspec`.

3.1.2 Performance Challenges

While `asyncssh` solved the permissions problem, it introduced an additional problem: the process was slow, especially when handling a large number of files. For example, processing 1000 files sequentially took approximately 3 minutes, which was unacceptable for a web application.

To improve the performance bottleneck, I experimented with alternative methods:

Parallel Requests with Async IO `asyncio` is a Python library that allows us to run single-threaded concurrent code inside coroutines, which are special functions that can pause and resume

their execution. Its method `asyncio.gather` allowed me to run multiple coroutines at once, hence handle concurrent file requests. With this, I significantly improved the processing time. The time dropped from around 3 minutes to 1.2 minutes for a path with 1760 objects. Although this was very efficient, it did not scale well with multiple users, and the scrum master, who is also the cluster administrator, noted the concern about overloading the cluster.

Bash Parallelization Instead of writing simple bash scripts, I decided to create a comprehensive script that would be executed on the cluster. This involved creating a loop that scans the ACL and POSIX permissions of each file in the cluster. By doing this, I shifted the computational load from the backend to the cluster. This improvement meant that I could send a script and only receive the results back, eliminating the need for post-processing as the list of authorized files would be directly sent to the frontend.

Initially, I considered further speeding up the process by running the program in parallel, similar to what was achieved with Python's `asyncio`. This led me to explore two tools: GNU Parallel and the POSIX standard command, `xargs`. Although GNU Parallel is powerful and easy to use, it is not available by default on all clusters. Therefore, I opted for `xargs`, which, despite creating fewer parallel processes (maximum nine in our case), offered compatibility and ease of integration with existing UNIX systems.

3.2 Implementation Challenges with `fsspec` and `Paramiko`

After the clone implementation, I also noticed that the `fsspec` implementation for creating SSH and SFTP connections was using `Paramiko` under the hood. This was inconvenient because the original SSH/NAS connector already used the `Paramiko` library, which had several drawbacks:

- **Performance Issues:** `Paramiko` is not optimized for high concurrency, leading to slower connection times and data transfer rates.
- **Scalability Concerns:** Handling a large number of simultaneous connections with `Paramiko` can cause significant performance degradation.
- **Complex Error Handling:** `Paramiko` has less intuitive error handling mechanisms, making it harder to debug and maintain.

Given these issues, there was a need for a new implementation. I considered either building one myself or find an existing solution. As mentioned earlier, `asyncssh` is particularly well-suited for applications requiring high concurrency and low latency, making it a preferable choice over `Paramiko`.

3.3 Discovery and Implementation of `sshfs`

During our exploration for alternatives to the solutions we had, we found `sshfs`, an implementation of `fsspec` for the SFTP protocol using `asyncssh`. `sshfs` provided several notable advantages over our previous approaches:

- **A Complete Implementation of the `fsspec` Protocol:** `sshfs` fully implements the `fsspec` protocol through SFTP, ensuring comprehensive support of file system operations. This would ensure consistency with the unified connector down the line.
- **Performance:** `sshfs` is quite fast compared to the alternative use case of `Paramiko` due to its asynchronous nature.
- **Ease of Use:** Since it is built on top of `fsspec`, it allows seamless integration into the existing architecture.

3.4 Development of the Unified Connector

The development of a unified connector for Madbot was a critical aspect of my internship. This unified connector aimed to streamline the process of linking external data sources, providing a consistent and reliable interface for the Madbot client.

Building upon the successful implementation of the SSH clone, I simplified the workload of creating the unified connector. The unified connector integrates the various connections as illustrated in Figure 2, ensuring a consistent and homogeneous result is returned to the client regardless of the backend format. This unification is crucial for maintaining a streamlined and user-friendly interface.

To enhance performance, the unified connector leverages the improvements achieved during the SSH clone implementation. By using asynchronous SSH (`asyncssh`) and parallel processing techniques, the connector efficiently handles multiple connections and data retrieval tasks concurrently. This ensures that the system remains responsive and scalable, even under heavy load.

Furthermore, the unified connector addresses the performance and scalability issues identified with `Paramiko`. By adopting `asyncssh`, the connector benefits from improved concurrency and reduced latency, which are essential for high-performance applications.

4 Discussion

4.1 Advances Made

During my internship, I successfully developed and implemented a new universal connector using the fsspec library. This development included the integration of sshfs to address the limitations posed by the existing SSH connector built on Paramiko. The primary improvements enabled by these results include:

- Enhanced permission verification processes, ensuring secure and reliable data access.
- Improved data retrieval capabilities, especially in handling large numbers of files.
- Significant performance enhancements through the implementation of asynchronous processes, which reduced processing times.
- Adoption of a consistent and reliable interface for data access within Madbot, facilitating better data management and integration.

4.2 Limitations

Despite the improvements, several limitations were encountered during the project:

- **Performance Bottlenecks with asyncssh**: While asyncssh provided faster performance, it posed a risk of overloading the cluster when handling multiple concurrent users.
- **Limitations of the Paramiko Library**: The existing SSH connector based on Paramiko had several drawbacks, including performance issues and complex error handling.
- **Complexity of Bash Scripts**: Reliance on bash scripts for permission verification introduced an additional layer of complexity that could impact maintainability and troubleshooting.
- **Parallel Process Limitations**: The adoption of XARGS over GNU Parallel, while necessary for compatibility, resulted in fewer parallel processes, which might not fully utilize the cluster's capabilities.
- **Additional Overhead for Permissions**: Running a separate connection to the cluster for permission checks added an overhead, as fsspec does not support running commands through SSH directly.

4.3 Future Work

Future work should focus on several key areas to further enhance the functionality and performance of the unified connector:

- **Optimization of Parallelization Strategy**: Further refine the parallelization strategy to better balance the load across the cluster without compromising performance. This could involve exploring alternative scripting tools that offer greater scalability.
- **Expanding Integration Capabilities**: Expand the functionality of the unified connector to support additional protocols and data sources, enhancing its versatility and applicability in various bioinformatics contexts.
- **Advanced Error Handling and Troubleshooting**: Investigate the integration of advanced error handling mechanisms and automated troubleshooting tools to improve the robustness and user experience of the connector.
- **Functional Testing**: Develop and implement functional tests to ensure the reliability and stability of the connector, thereby enhancing its usability for researchers.
- **Continued Integration Improvements**: Continue to improve the integration and write additional functional tests to ensure the connector's reliability and performance.

4.4 Conclusion

In conclusion, the development and testing of the SSH connector within the Madbot project have demonstrated significant advancements in data management and integration. The implementation of asynchronous processes and the adoption of the fsspec library have resulted in improved performance and scalability. Despite the challenges encountered, the outcomes of this internship provide a solid foundation for future enhancements and integrations, paving the way for a more effective and comprehensive data management solution in madbot and bioinformatics at large.The unified connector represents a significant advancement in the Madbot infrastructure. It not only consolidates various data connections into a single, coherent interface but also enhances performance and scalability. This development is expected to facilitate more efficient data integration and retrieval processes for Madbot users, ultimately contributing to the overall success of the platform.

References

[Allan et al., 2012] Allan, C., Burel, J.-M., Moore, J., Blackburn, C., Linkert, M., Loynton, S., MacDonald, D., Moore, W. J., Neves, C., Patterson, A., Porter, M., Tarkowska, A., Loranger, B., Avondet, C., Lagerstedt, I., Lianas, L., Leo, S., Hands, K., Hay, R. T., Patwardhan, A., Best, C., Kleywegt, G. J., Zanetti, G., and Swedlow, J. R. (2012). Omero: flexible, model-driven data management for experimental biology. *Nature Methods*, 9(3):245–253.

[Beck et al., 2001] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. (2001). Manifesto for Agile Software Development. Accessed: 2024-06-20.

[BioData Inc., 2024] BioData Inc. (2024). Labguru laboratory management software. `https://www.labguru.com`. Accessed: 2024-06-19.

[Deutz et al., 2020] Deutz, D., Buss, M., Hansen, J., Hansen, K., Kjelmann, K., Larsen, A., Vlachos, E., and Holmstrand, K. (2020). How to fair: a danish website to guide researchers on making research data more fair.

[Django Software Foundation, 2023] Django Software Foundation (2023). Django (version 5.0.3) [computer software]. Lawrence, Kansas.

[Durant, 2024] Durant, M. (2024). fsspec: Filesystem specification. Accessed: 2024-06-18.

[ELIXIR, 2024] ELIXIR (2024). Elixir: A distributed infrastructure for european biological data. `https://elixir-europe.org/about-us/who-we-are/nodes/france`. Accessed: 2024-06-19.

[Rajasekar et al., 2010] Rajasekar, A., Moore, R., Hou, C.-Y., Lee, C., Marciano, R., de Torcy, A., Wan, M., Schroeder, W., Chen, S.-Y., Gilbert, L., Tooby, P., and Zhu, B. (2010). *iRODS Primer: Integrated Rule-Oriented Data System*, volume 2 of *Synthesis Lectures on Information Concepts, Retrieval, and Services*. Morgan & Claypool Publishers.

[The Galaxy Project Team, 2024] The Galaxy Project Team (2024). Galaxy project: Enabling accessible, reproducible, and transparent computational biomedical research. `https://galaxyproject.org`. Accessed: 2024-06-19.

[Yuan et al., 2023] Yuan, D., Ahamed, A., Burgin, J., Cummins, C., Devraj, R., Gueye, K., Gupta, D., Gupta, V., Haseeb, M., Ihsan, M., Ivanov, E., Jayathilaka, S., Kadhirvelu, V. B., Kumar, M., Lathi, A., Leinonen, R., McKinnon, J., Meszaros, L., O'Cathail, C., Ouma, D., Paupério, J., Pesant, S., Rahman, N., Rinck, G., Selvakumar, S., Suman, S., Sunthornyotin, Y., Ventouratou, M., Vijayaraja, S., Waheed, Z., Woollard, P., Zyoud, A., Burdett, T., and Cochrane, G. (2023). The European Nucleotide Archive in 2023. *Nucleic Acids Research*, 52(D1):D92–D97.

[Zenodo, 2024] Zenodo (2024). Zenodo: Open science at cern. `https://openscience.cern/zenodo`. Accessed: 2024-06-19.