Chandrasekhar *et al.*
Journal of Cheminformatics (2023) 15:98
https://doi.org/10.1186/s13321-023-00762-4

Journal of Cheminformatics

SOFTWARE

Open Access

Cheminformatics Microservice: unifying access to open cheminformatics toolkits

Venkata Chandrasekhar[1], Nisha Sharma[1], Jonas Schaub[1], Christoph Steinbeck[1] and Kohulan Rajan[1*]

Abstract

In recent years, cheminformatics has experienced significant advancements through the development of new open-source software tools based on various cheminformatics programming toolkits. However, adopting these toolkits presents challenges, including proper installation, setup, deployment, and compatibility management. In this work, we present the Cheminformatics Microservice. This open-source solution provides a unified interface for accessing commonly used functionalities of multiple cheminformatics toolkits, namely RDKit, Chemistry Development Kit (CDK), and Open Babel. In addition, more advanced functionalities like structure generation and Optical Chemical Structure Recognition (OCSR) are made available through the Cheminformatics Microservice based on pre-existing tools. The software service also enables developers to extend the functionalities easily and to seamlessly integrate them with existing workflows and applications. It is built on FastAPI and containerized using Docker, making it highly scalable. An instance of the microservice is publicly available at https://api.naturalproducts.net. The source code is publicly accessible on GitHub, accompanied by comprehensive documentation, version control, and continuous integration and deployment workflows. All resources can be found at the following link: https://github.com/Steinbeck-Lab/cheminformatics-microservice.

Keywords CDK, RDKit, Open Babel, Cheminformatics, Toolkits, Microservice

Graphical Abstract

*Correspondence:
Kohulan Rajan
kohulan.rajan@uni-jena.de
[1] Institute for Inorganic and Analytical Chemistry, Friedrich Schiller University Jena, Lessingstr. 8, 07743 Jena, Germany

Introduction

Open cheminformatics toolkits, large-scale chemical databases, and an increase in available computing power have led to significant advancements in the field of cheminformatics in recent years [1, 2]. As a result,

large amounts of chemical data can be handled and analysed efficiently, which in turn benefits research fields like chemistry, drug discovery, and material design. Multiple prominent open-source cheminformatics toolkits are available, which include RDKit [3], Open Babel [4], Chemistry Development Kit (CDK) [5, 6], Indigo [7], and the recently developed Python-based Informatics Kit for Analysing CHemical Units (PIKAChU) [8]. A summary of their native programming language, latest version, and licence information is given in Table 1.

All cheminformatics toolkits mentioned above offer ready-to-use routines for common tasks like data format conversions, descriptor calculations, and structure editing. On top of these, every toolkit has an individual set of more advanced functionalities like coordinate generation, substructure analysis, or structure normalisation. For this reason, researchers often use multiple toolkits in their tools or workflows [9, 10]. To do so, they have to familiarise themselves with the specific requirements, syntax, and algorithms of each employed toolkit. Being familiar with the underlying programming languages such as Python, Java, or C++ is required to achieve this most efficiently. It is also essential to have a thorough understanding of chemical concepts, molecular representations, and computational algorithms.

For developing cheminformatics workflows, machine learning models, or web applications, researchers and software developers need to set up a proper working environment for the toolkits in order to integrate them into their applications. These software tools can later become cumbersome to use and maintain if not set up properly or inadequately documented. Unfortunately, this is often the case due to the complexity of the setup, lack of documentation, and lack of good research data management practices [11]. Other challenges in developing cheminformatics software, databases, and web applications result from building on top of these toolkits due to factors such as various toolkit version management [12], dependencies management [13], and maintenance [14].

- Software Version Management: The purpose of this process is to effectively manage the versions of software and toolkits throughout their development and maintenance cycle. The objective is to organise and track changes, facilitate collaboration, and ensure the stability and integrity of software projects, thus increasing productivity and streamlining development processes. This process can be time-consuming and requires careful planning and organisation.
- Dependencies: The majority of cheminformatics tools require several interdependent third-party libraries to function. Managing these dependencies can be challenging since developers must ensure that each dependency is installed correctly and compatible with the other dependencies. Otherwise, this can lead to conflicts between dependencies, which may cause the software to malfunction.
- Maintenance: It is challenging to maintain software and databases as they require regular updates and bug fixes to remain up-to-date and functional. In the special case of open-source software, this usually requires a large community of active and committed users and developers.

Most cheminformatics open-source programming toolkits have a rather high entrance barrier due to the abovementioned aspects. Also, the quality of available documentation and tutorials varies. This is especially critical for young researchers new to the field who have to spend a lot of setup time before being able to work on their research projects. Therefore, online tools for cheminformatics are becoming increasingly popular due to their usual ease of use, adequate documentation, and the fact that no or only little programming knowledge is required to use them [15–18]. The use of web-based solutions or solutions that are driven by Application Programming Interfaces (APIs) [19] offers a lower entrance barrier. In addition, these services are platform-independent and can be easily integrated with software utilities, databases, repositories, and cheminformatics data management workflows [17].

Table 1 Summary of widely used open-source cheminformatics toolkits

Toolkit	Programming language	Latest version	Licence	Source code
Open Babel	C++	3.1.1	GNU general public license (GPL)	https://github.com/openbabel/openbabel
CDK	Java	2.8.0	GNU lesser general public license v2.1	https://github.com/cdk/cdk
RDKit	C++, Python	2023.03.2	BSD 3-clause license	https://github.com/rdkit/rdkit
PIKAChU	Python	1.0.13	MIT license	https://github.com/BTheDragonMaster/pikachu
Indigo	C/C++	1.11.0	Apache license 2.0	https://github.com/epam/Indigo

To overcome the challenging integration of third-party libraries like cheminformatics toolkits into application code, two common software development techniques can be used: containerization and microservices. Microservices [20], also referred to as microservice architecture, is a collection of small, autonomous services that can be deployed, scaled, and maintained individually [21]. By leveraging well-defined APIs, each microservice carries out a specific function and interacts with other microservices. These services are characterised by their granular nature and employ lightweight protocols to enable the independent execution of each service [22]. The native installation of such services on a system makes maintenance difficult in the long run. The service may not function as a result of Operating System (OS) level updates or software environment conflicts. In addition, in the event of a service failure, it is necessary to reboot the entire system. In order to address these issues, containerization can be used for the deployment of software services and applications. Containers are lightweight isolated environments that enable applications and their dependencies to run consistently across a variety of systems, independent of the OS [23]. In addition, a container provides a consistent and reproducible execution environment across development, testing, and production environments. In this work, containerization was achieved using Docker [24]. Software components can be containerized using Docker and distributed publicly via the Docker Hub [25], a cloud-based registry provided by Docker that allows developers to store, share, and distribute Docker images.

This article presents the Cheminformatics Microservice, an open-source solution for handling chemical data and performing various cheminformatics tasks by employing multiple cheminformatics toolkits (CDK, RDKit, and Open Babel). These tasks include generating high-quality chemical structure depictions and 3D conformers, calculating molecular descriptors and IUPAC names, and converting SMILES representations of chemical structures into other machine-readable formats. The microservice can be accessed through a unified REST (REpresentational State Transfer) [26] interface, which is made available as a public server for anyone to access via https://api.naturalproducts.net/. Alternatively, it can be installed locally or on a private server using the provided Docker image. It can also be deployed and auto-scaled on a Kubernetes-managed private cluster in just a few steps via the Helm Charts provided [27]. These deployment options are designed to be user-friendly since they require no prior knowledge of the underlying toolkits or their setup environment. They make the presented software service suitable for a wide range of applications in academic and industrial environments. The entire Cheminformatics Microservice source code is made available to the public on GitHub: https://github.com/Steinbeck-Lab/cheminformatics-microservice. Users and researchers are encouraged to submit feature requests and contribute to the microservice to ensure its continued growth.

Implementation

Cheminformatics Microservice is developed using FastAPI, a web framework for generating RESTful APIs with Python. FastAPI was chosen for this project due to its speed, efficiency, and suitability for building advanced APIs. It enables the straightforward creation of robust and scalable APIs. Docker is used for containerization, and semantic versioning principles are applied to track code changes and toolkit versions. In the microservice container, the cheminformatics toolkits RDKit and Open Babel are accessed natively using Python, while the Chemistry Development Kit (CDK) is integrated using JPype [28]. Cheminformatics Microservice consists of five modules, namely *chem*, *convert*, *depict*, *ocsr*, and *tools*. Compliant with the OpenAPI [29] specification version 3.1.0, this work provides standard documentation, encourages interoperability, enables automatic code generation, simplifies validation, and integrates with a variety of tools and libraries to enhance the functionality of REST (REpresentational State Transfer) APIs. An overview of the software architecture is given in Fig. 1.

The *chem* module offers various functions such as descriptor calculation, stereoisomer enumeration, HOSE code [30] generation, NPLikeness score calculation [31], ClassyFire classification [32], molecular structure standardisation [33], and a preprocessing pipeline for the upcoming version of the COCONUT database as explained in the results section below. The *convert* module provides conversions from SMILES [34] to other molecular string representations such as InChI [35, 36], InChIKey [36], canonical SMILES [37], CXSMILES [38], SELFIES [39, 40], and IUPAC names. The latter is achieved via the Smiles TO iUpac Translator (STOUT) version 2.0 toolkit [41]. Additionally, MOL files can be generated with 2D and 3D coordinates from SMILES input. Using the *depict* module, one can generate 2D depictions of chemical structures with various settings, including an option to generate 2D representations with stereochemical annotations following the Cahn–Ingold–Prelog [42] (CIP) sequence rules. RDKit or CDK can be used to generate 2D representations. The depictions are generated in an SVG format and may be scaled to fit the needs of the user. It is also possible to generate a 3D depiction [43] using this module, which is useful as a chemical structure display option for databases or as a teaching aid. The underlying 3D conformer is generated using RDKit. The *ocsr* module incorporates Deep lEarning for Chemical ImagE Recognition (DECIMER)

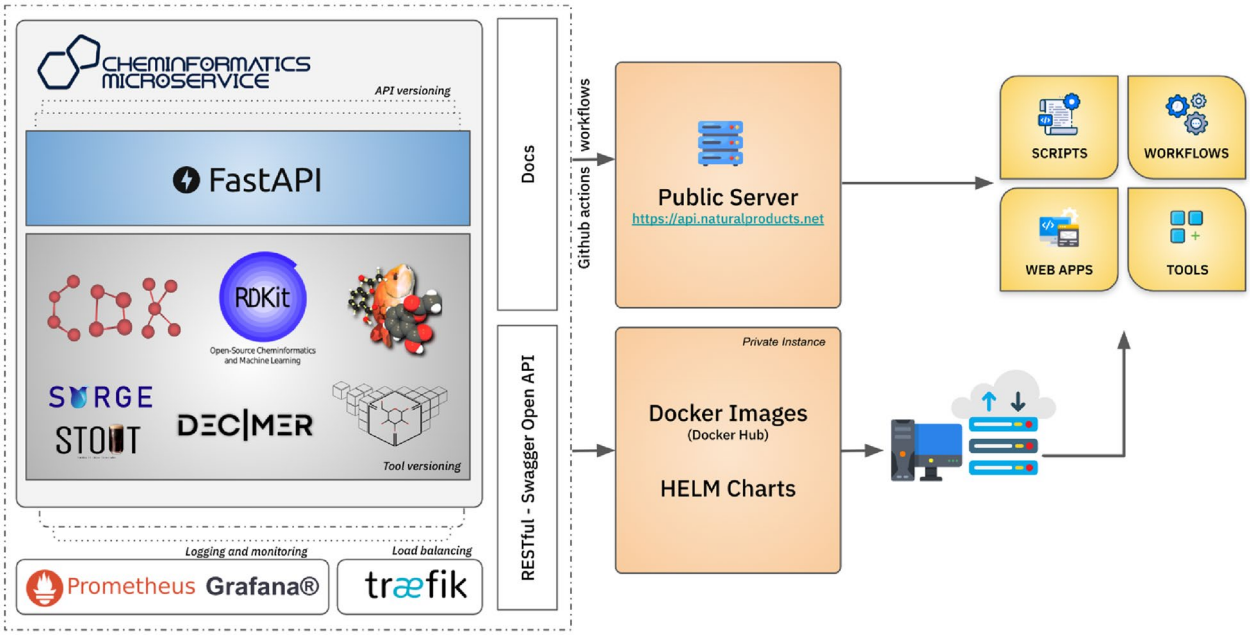

Fig. 1 Overview of the Cheminformatics Microservice public server architecture. The public server can be accessed via https://api.naturalproducts.net/latest/docs

modules [44, 45] for translating images of chemical structures into machine-readable SMILES representations. These can be accessed via HTTP *POST* requests. Finally, the *tools* module is designated as a miscellaneous collection of advanced cheminformatics tools. It offers a function to generate chemical structures from a molecular formula given as input using the surge chemical graph generator [46]. Other functions of the *tools* module can be used to identify glycosidic moieties in input molecules and to remove them in order to generate the aglycone structure. These routines are implemented based on the Sugar Removal Utility (SRU) [47].

The functionalities offered by each main module described above can also be implemented via independent Python functions, enabling users to access Cheminformatics Microservice natively without the REST interface. Users can import and use it like any other package directly in their own Python code. Individual toolkit wrapper modules provide access to the three cheminformatics toolkits RDKit, CDK, and Open Babel. RDKit and Open Babel are natively accessible, while CDK, a Java library, has functionalities ported to Python using JPype. It is possible to extend the functionalities provided by cheminformatics toolkits in the future by using these wrapper modules. Separating them into individual modules is necessary to achieve granular control over the functions. This also ensures that the entire system will not be broken if one module is affected by potential software failures. The Python functions are documented separately, and the documentation can be accessed via: https://cheminformatics-microservice.readthedocs.io/en/latest/.

To ensure reproducibility, a consistent versioning system is essential. Best practices for research data management [48, 49] recommend documenting software and component versions separately, especially for tools like the presented microservice with multiple dependencies. Cheminformatics Microservice uses multi-level versioning to record API and software dependencies. The codebase undergoes bi-annual major releases, with corresponding documentation provided for the underlying toolkits, tools, and environment dependencies for each release. The API version updates are released only when significant changes to the API endpoints have been made. It is possible to update the underlying cheminformatics toolkits whenever new releases are published without having to update the entire code base since the REST API remains unchanged. The API usage can be logged, monitored, and visualised using Prometheus [50] and Grafana [51] in a standalone or distributed environment. Cheminformatics Microservice can also be deployed using a Continuous Integration and Continuous Deployment (CI/CD) pipeline via GitHub Actions [52]. Code integration, testing, and application deployment are automated using CI/CD, which fosters collaboration, minimises manual tasks, and enables timely feedback.

Results and discussion

The presented Cheminformatics Microservice provides straightforward access to the open-source cheminformatics toolkits RDKit, CDK, and OpenBabel, as well as deep learning-based tools such as Deep lEarning for Chemical ImagE Recognition (DECIMER) for OCSR and Smiles TO iUpac Translator (STOUT). Building on top of these tools and toolkits, it makes a diverse selection of important functionalities needed by cheminformaticians on a daily basis accessible via a RESTful interface. Additionally, Cheminformatics Microservice includes a number of widely used packages, like the ChEMBL [53] curation pipeline [33], CDK-based sugar removal functionalities [47], and the open-source structure generator surge [46]. The presented microservice is intended to facilitate the handling of large amounts of chemical structural data to enable the development of adaptable, scalable, and maintainable cheminformatics applications.

The main modules of Cheminformatics Microservice are well-documented and can be accessed via the following link: https://api.naturalproducts.net/. In order to obtain the output generated by the microservice, each API module uses either a *GET* or a *POST* HTTP request method [54]. Most of the services provided by the *chem, convert,* and *depict* modules can be accessed using SMILES as an input format for molecular structures. Where a specific functionality provided by the microservice can be achieved in a similar manner with RDKit, CDK, or Open Babel, the user has the option of employing the toolkit of their choice via an additional parameter. The *chem* module offers routines for structure manipulation and standardisation, descriptor calculation, and chemical classification. The *standardize* functionality is used via a *POST* method for the purpose of standardising molecules represented by MOL format tables through the ChEMBL structure curation pipeline. The *convert* module enables users to convert SMILES representations into other formats of their choice, using the *GET* method. Meanwhile, the *depict* module allows users to generate customised 2D depictions of molecular structures, offering options for coloured or black and white output images. The stereochemical annotations on the 2D depictions are generated using the Cahn-Ingold-Prelog (CIP) priority rules. To accomplish this, the Java package *centres* [55, 56] is used in conjunction with CDK. Additionally, the *depict* module is able to produce interactive 3D models of input structures. Figure 2A, B illustrates the application of the depict module to generate a 2D depiction with CIP annotations in colour on a scale of 512 × 512 pixels

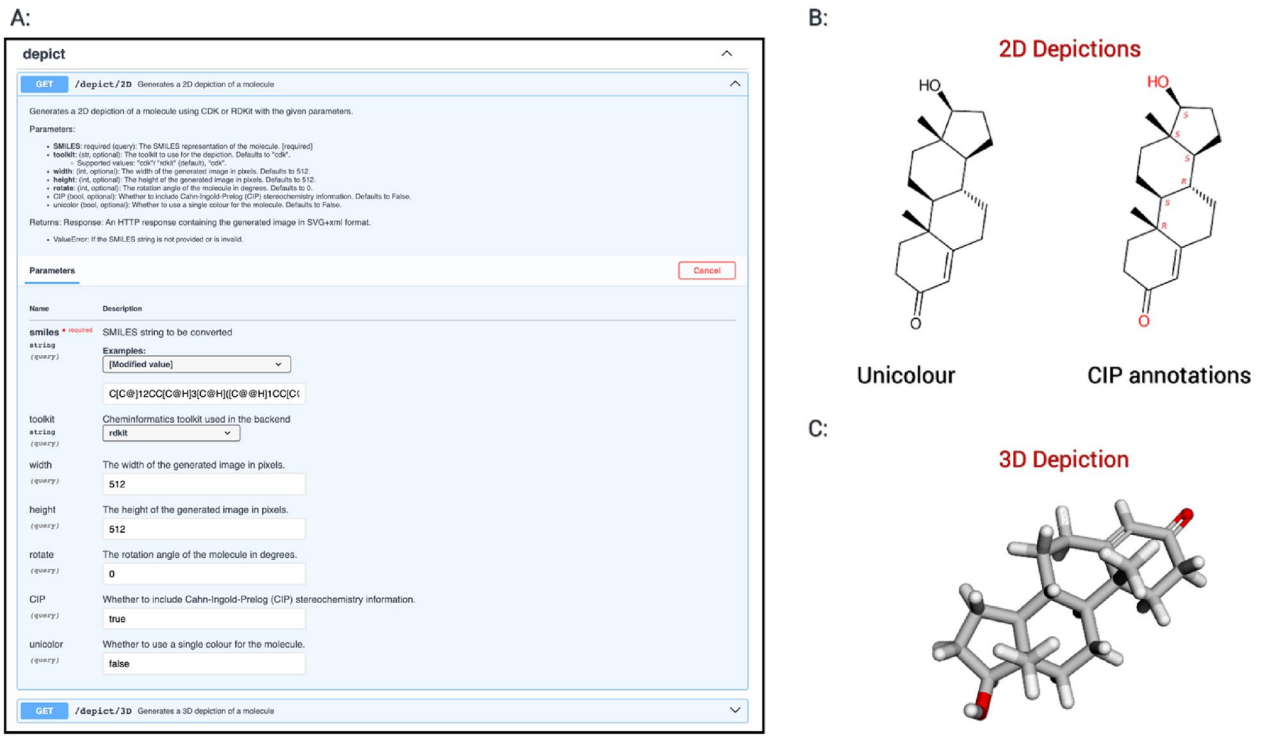

Fig. 2 **A** The screenshot showcases the output of the module, displaying the depiction options for the input molecule Testosterone given as a SMILES string. **B** The output of the *depict* module with unicolor and CIP annotations for 2D depictions, and **C** the interactive 3D depiction [59]

and with a 52° rotation. The depicted image with CIP annota*depict*tions can be generated directly by using this call to the API: https://api.naturalproducts.net/latest/depict/2D?smiles=C%5BC@%5D12CC%5BC@H%5D3%5BC@H%5D(%5BC@@H%5D1CC%5BC@@H%5D2O)CCC4=CC(=O)CC%5BC@%5D34C&width=512&height=512&rotate=52&&CIP=true&unicolor=false&toolkit=cdk

2D representations are produced in the form of SVG images, whereas 3D representations are returned as HTML files containing embedded JSMol objects [57, 58].

Through REST API calls, the *tools* and *ocsr* modules offer convenient access to the underlying functionalities of the integrated tools. For example, the *tools* module allows users to employ the open-source chemical structure generator surge to generate chemically valid structures based on a provided molecular formula. The generated structures are returned as a list of SMILES representations. In order to address the resource-intensive nature of the chemical structure generation process, a maximum limit of 10 heavy atoms in the given molecular formula has been imposed. This restriction applies exclusively to the public instance. The *tools* module also allows users to access the Sugar Removal Utility (SRU) to detect and remove linear and circular sugar moieties in/from input structures. Users can access the DECIMER toolkit through the *ocsr* module, which enables identification, segmentation, and translation into machine-readable representations of chemical structure depictions from the scientific literature.

Currently, Cheminformatics Microservice is available to the public via https://api.naturalproducts.net/latest/docs and in the back end, the container is running on a compute server with the processor Intel(R) Xeon(R) Gold 6226R CPU @ 2.90 GHz and 16 GB of RAM.

Use of Cheminformatics Microservice in the COCONUT database

Cheminformatics Microservice is extensively employed in the upcoming version of the COCONUT (COlleCtion of Open Natural prodUcTs) database that is currently under development. With the *depict* module, it becomes feasible to present all the natural product structure data entries within COCONUT in both 2D and 3D formats (Fig. 3). The microservice also includes the generation of molecular descriptors and further preprocessing steps for data submission to the COCONUT database.

Documentation

Cheminformatics Microservice offers detailed documentation alongside its source code, ensuring that users can easily access and navigate it without a high entry barrier. The documentation provides clear guidance on how to effectively use, deploy, and install the software. Figure 4 offers a glimpse of the documentation that is deployed using GitHub Pages and can be accessed at the following URL:

https://docs.api.naturalproducts.net.

Performance and scalability

Scaling is particularly important for those who plan to use the API endpoints for database generation, large-scale descriptor calculation, or automated literature mining since it considerably reduces the overall computation time required. Cheminformatics Microservice is specifically designed to scale at large, in a local deployment as well as an orchestrated cluster. It can be configured to run on multiple workers when deployed independently. If deployed through Docker Compose or over a Kubernetes cluster, the microservice can be auto-scaled infinitely to handle incoming requests. Helm Chart or the Docker Compose file provided in the codebase enable scaling

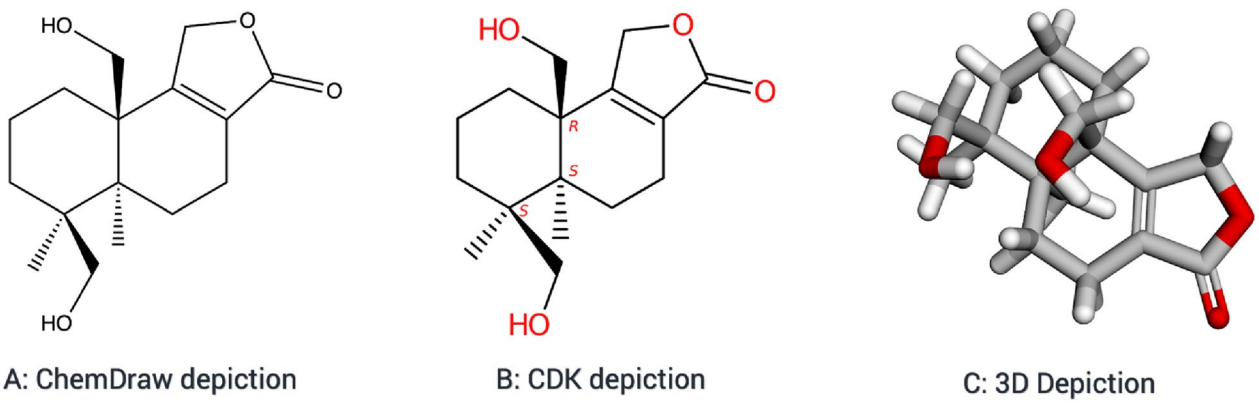

Fig. 3 The molecule Astellolide W [60] was extracted from the referenced article using DECIMER. The structure is depicted using ChemDraw (**A**) and CDK (**B**) for 2D representations, while the 3D depiction (**C**) was created using the presented microservice

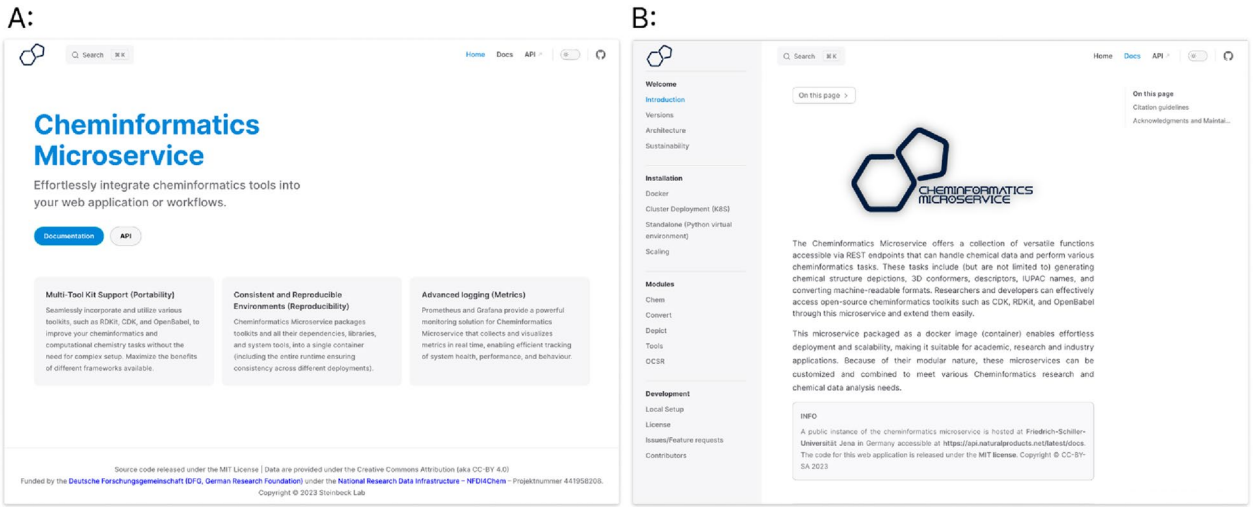

Fig. 4 Screenshots of the Cheminformatics Microservice landing page (**A**) and detailed documentation (**B**)

without any additional setup. Prometheus and Grafana are used to monitor the request count over time. These logs and other performance indicators will facilitate the scaling of Cheminformatics Microservice workers based on user demand in the future.

To determine scalability, stress testing was performed using Vegeta [61]. To determine the maximum throughput Cheminformatics Microservice can handle when deployed on a machine with Intel(R) Xeon(R) Gold 6226R CPU @ 2.90 GHz and 16 GB of RAM, requests in small increments were added using Vegeta and the delivered throughput was measured until a limit was reached (Fig. 5). This stress test indicated that the service reached its maximum capacity when handling approximately 2500 requests per second for echo requests, after which the success rate began to decline. When queried with the task to generate 2D coordinates for a molecular structure given as a SMILES string using RDKit, Fig. 5 clearly demonstrates that the microservice can effectively handle a wide range of requests, ranging from 50 to 500 per second. With an increase in the input molecule size and number of requests per second, this rate starts to deteriorate. A detailed description of the stress test can be found in the Additional file 1.

Nevertheless, the default configuration of Cheminformatics Microservice ensures excellent stability in handling user requests and effectively manages to execute the computation required.

However, limitations are imposed on the public instance for some specific tools and routines. For example, restrictions are imposed in the structure generator surge when dealing with molecules containing more than ten heavy atoms, as this service demands significant computational resources. Access to mass mining through the DECIMER endpoint is also restricted. These restrictions might be reconsidered in the future, depending on the user demand and computation resource availability.

Conclusion

The presented Cheminformatics Microservice is a self-contained, web-based service that operates independently of the operating system. This can be used by researchers with no or limited programming experience to perform daily cheminformatics tasks effortlessly via the API hosted at https://api.naturalproducts.net. This service allows a diverse range of open-source cheminformatics toolkits to be accessed without requiring any software installation or environment setups. To the best of our knowledge, it is currently the sole microservice in the field of cheminformatics offering users the ability to access multiple toolkits, as well as additional tools such as the open-source structure generator surge, Sugar Removal Utility (SRU), and the DECIMER OCSR tools. Cheminformatics Microservice is designed to be user-friendly, easily extendable, deployable, and scalable. It can be accessed through the public API or hosted on private clusters or single machines.

The integration of multiple cheminformatics toolkits in a centralised platform enhances user accessibility to cheminformatics utilities. By using industry-standard monitoring, deployment, documentation, and code quality standards, this project demonstrates software development in a user-focused manner. By making the source code and the documentation completely open and public, we aim to make valuable contributions to the scientific community while also enabling the submission of feature

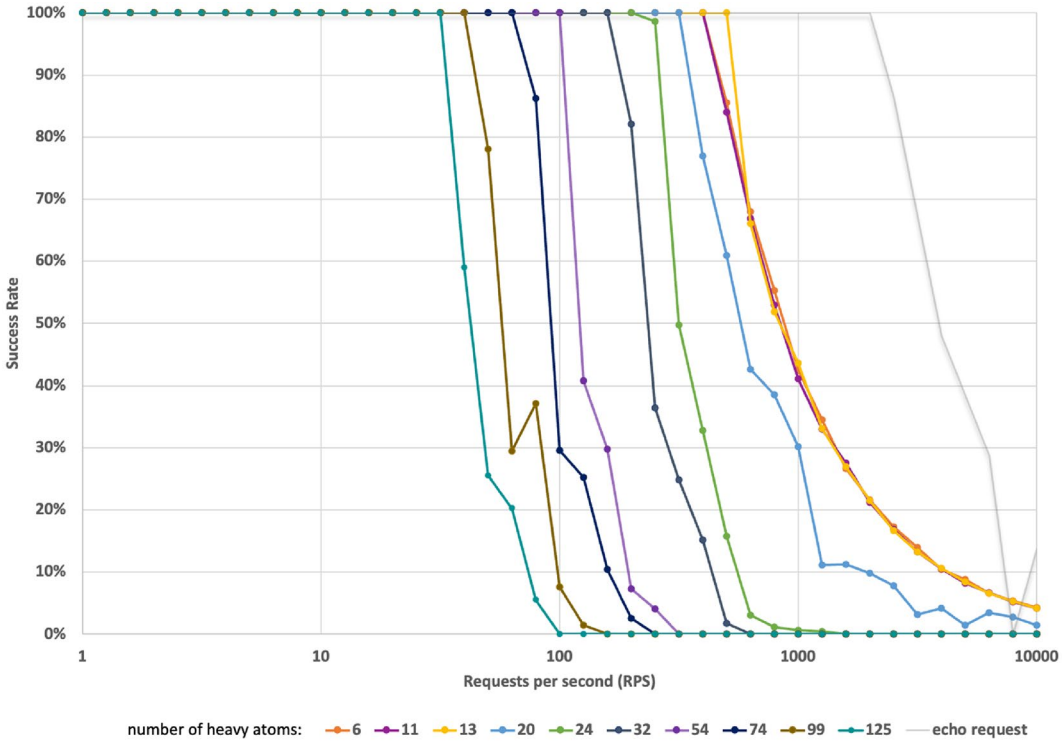

Fig. 5 A snapshot of performance comparing the success rate with an increase in the number of requests per second as well as with an increase in the number of heavy atoms in the input. The grey line represents the maximum number of requests the server could process (echo request)

requests for future enhancements. Cheminformatics Microservice is anticipated to become an invaluable asset for cheminformaticians due to its inherent expandability. It follows a clear semantic versioning system, receives bi-annual updates, and includes comprehensive documentation. These characteristics significantly facilitate data reproduction and reuse for researchers, thereby fostering better collaboration among peers.

Availability and requirements

- Project name: Cheminformatics Microservice
- Project home page: https://github.com/Steinbeck-Lab/cheminformatics-microservice
- Docker Image: https://hub.docker.com/r/nfdi4chem/cheminformatics-microservice
- Helm Chart repo: https://nfdi4chem.github.io/repo-helm-charts/
- Helm Chart GitHub: https://github.com/NFDI4Chem/repo-helm-charts
- Current version: v1.6.0
- DOI of archived current release: https://doi.org/10.5281/zenodo.7745987
- Operating system(s): Independent
- Programming language: Python 3, HTML
- Requirements:

API calls:

Internet connection and command line interface or a web browser

Run locally:

Docker—To use Cheminformatics Microservice as a Docker container.
Conda environment—to use Cheminformatics Microservice natively without Docker.

Dependencies (managed by Docker/Conda):

Python packages: uvicorn $\geq 0.15.0, < 0.16.0$, fastapi $\geq 0.80.0$, fastapi-pagination $== 0.10.0$, fastapi-versioning $\geq 0.10.0$, prometheus-fastapi-instrumentator, jpype1 $== 1.4.1$, jinja2, pandas, chembl_structure_pipeline, HOSE_code_generator @ git + https://github.com/Ratsemaat/HOSE_code_generator, websockets $== 10.4$, pillow $== 9.4.0$, opencv-python $== 4.7.0.68$,

matplotlib = = 3.4.3, scikit-image, pdf2image = = 1.16.2, IPython, pystow ≥ 0.4.9, unicodedata2 = = 15.0.0, efficientnet, tensorflow = = 2.12.0, pillow-heif = = 0.10.0, selfies ≥ 2.1.1, httpx ≥ 0.24.1, keras_preprocessing = = 1.1.2, decimer-segmentation ≥ 1.1.2, STOUT-pypi ≥ 2.0.5 and decimer ≥ 2.2.0

Java: OpenJDK for Java 11

Java Libraries: CDK 2.8.0, SRU 1.3.2 and Centres 1.0

- Licence: MIT
- Documentation:

Home page: https://docs.api.naturalproducts.net/
API: https://api.naturalproducts.net/latest/docs
Python Documentation: https://cheminformatics-microservice.readthedocs.io/en/latest/

- Any restrictions to use by non-academics: None.

Abbreviations

2D	Two dimensional
3D	Three dimensional
APIs	Application Programming Interfaces
BSD	Berkeley Software Distribution
CC	Creative Commons
CDK	Chemistry Development Kit
CI/CD	Continuous Integration and Continuous Deployment
CIP	Cahn–Ingold–Prelog
COCONUT	COlleCtion of Open Natural prodUcTs
CPU	Central processing unit
CXSMILES	Chemaxon Extended Simplified Molecular Input Line Entry Specification
DECIMER	Deep lEarning for Chemical ImagE Recognition
GB	Gigabyte
GHz	Gigahertz
GNU	GNU's Not Unix
GPL	General Public License
HOSE	Hierarchically ordered spherical environment
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
InChI	International Chemical Identifier
IUPAC	International Union of Pure and Applied Chemistry
JDK	Java Development Kit
JPype	Java for Python
JSMol	JavaScript molecular viewer
MIT	Massachusetts Institute of Technology
NPLikeness	Natural product likeness
OCSR	Optical chemical structure recognition
OS	Operating system
PC	Personal computer
PIKAChU	Python-based Informatics Kit for Analysing CHemical Units
RAM	Random access memory
REST	REpresentational state transfer
SELFIES	SELF-referencIng embedded strings
SMILES	Simplified molecular input line entry specification
SRU	Sugar Removal Utility
STOUT	SMILES-TO-IUPAC-name translator
SVG	Scalable Vector Graphics

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s13321-023-00762-4.

Additional file 1: Performance/Stress test results.

Acknowledgements
The authors would like to thank Prof. Dr Achim Zielesny for his valuable input throughout the development of Cheminformatics Microservice and Mr Henning Otto Brinkhaus for his assistance with the JPype implementation.

Author contributions
VC and KR initiated the project and developed the software. NS developed, automated, and documented the work. JS helped with Java porting. KR designed the logo. VC, JS, and KR wrote the paper together. KR and CS supervised the study. All authors read and approved the final manuscript.

Funding
Open Access funding enabled and organized by Projekt DEAL. Open Access funding enabled and organized by Projekt DEAL. This work was funded by the German Research Foundation under project number: 239748522- SFB 1127 ChemBioSys (Project INF) and NFDI4Chem under project number 441958208.

Data availability
Not Applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
The authors have given their consent for the work to be published.

Competing interests
The authors do not have any competing interests to declare.

Received: 7 July 2023 Accepted: 19 September 2023
Published: 16 October 2023

References

1. Ambure P, Aher RB, Roy K (2014) Recent advances in the open access cheminformatics toolkits, software tools, workflow environments, and databases. In: Zhang Wei (ed) Methods in pharmacology and toxicology. New York, Springer, pp 257–296
2. Wegner JK, Sterling A, Guha R, Bender A, Faulon J-L, Hastings J, O'Boyle N, Overington J, Van Vlijmen H, Willighagen E (2012) Cheminformatics. Commun ACM 55:65–75. https://doi.org/10.1145/2366316.2366334
3. Landrum G, et al. RDKit: open-source cheminformatics software. 2016. http://www.rdkit.org/. https://github.com/rdkit/rdkit. Accessed 10 July 2023
4. O'Boyle NM, Banck M, James CA, Morley C, Vandermeersch T, Hutchison GR (2011) Open babel: an open chemical toolbox. J Cheminform 3:33. https://doi.org/10.1186/1758-2946-3-33
5. Willighagen EL, Mayfield JW, Alvarsson J, Berg A, Carlsson L, Jeliazkova N, Kuhn S, Pluskal T, Rojas-Chertó M, Spjuth O et al (2017) The chemistry development kit (CDK) v2.0: atom typing, depiction, molecular formulas, and substructure searching. J Cheminf. https://doi.org/10.1186/s13321-017-0220-4
6. Steinbeck C, Han Y, Kuhn S, Horlacher O, Luttmann E, Willighagen E (2003) The chemistry development kit (CDK): an open-source java library for

chemo- and bioinformatics. J Chem Inf Comput Sci 43:493–500. https://doi.org/10.1021/ci025584y
7. Indigo Toolkit. https://lifescience.opensource.epam.com/indigo/. Accessed 25 June 2020.
8. Terlouw BR, Vromans SPJM, Medema MH (2022) PIKAChU: a python-based informatics kit for analysing chemical units. J Cheminform 14:34. https://doi.org/10.1186/s13321-022-00616-5
9. Brinkhaus HO, Rajan K, Zielesny A, Steinbeck C (2022) RanDepict: random chemical structure depiction generator. J Cheminform 14:31. https://doi.org/10.1186/s13321-022-00609-4
10. Zulfiqar M, Gadelha L, Steinbeck C, Sorokina M, Peters K (2023) MAW: the reproducible metabolome annotation workflow for untargeted tandem mass spectrometry. J Cheminform 15:32. https://doi.org/10.1186/s13321-023-00695-y
11. Ashiq M, Usmani MH, Naeem M (2022) A systematic literature review on research data management practices and services. Glob Knowl Mem Commun 71:649–671. https://doi.org/10.1108/gkmc-07-2020-0103
12. Van Gurp J, Prehofer C. Version management tools as a basis for integrating product derivation and software product families. In: Proceedings of the proceedings of the workshop on variability management-working with variability mechanisms at SPLC; 2006; pp. 48–58.
13. Esparrachiari S, Reilly T, Rentz A (2018) Tracking and controlling microservice dependencies. ACM Queue 16:44–65. https://doi.org/10.1145/3277539.3277541
14. Canfora G, Cimitile A (2001) Software maintenance. In: Chang SK (ed) Handbook of software engineering and knowledge engineering. World Scientific Publishing Company, Singapore, pp 91–120
15. Huang Y-C, Tremouilhac P, Nguyen A, Jung N, Bräse S (2021) ChemSpectra: a web-based spectra editor for analytical data. J Cheminform 13:8. https://doi.org/10.1186/s13321-020-00481-0
16. Jablonka KM, Moosavi SM, Asgari M, Ireland C, Patiny L, Smit B (2020) A data-driven perspective on the colours of metal-organic frameworks. Chem Sci 12:3587–3598. https://doi.org/10.1039/d0sc05337f
17. Patiny L, Borel A (2013) ChemCalc: a building block for tomorrow's chemical infrastructure. J Chem Inf Model 53:1223–1228. https://doi.org/10.1021/ci300563h
18. Patiny L, Zasso M, Kostro D, Bernal A, Castillo AM, Bolaños A, Asencio MA, Pellet N, Todd M, Schloerer N et al (2018) The C6H6 NMR repository: an integral solution to control the flow of your data from the magnet to the public. Magn Reson Chem 56:520–528. https://doi.org/10.1002/mrc.4669
19. Ofoeda J, Boateng R, Effah J (2019) Application programming interface (API) research. Int J Enterp Inf Syst 15:76–95. https://doi.org/10.4018/ijeis.2019070105
20. Newman S (2015) Building microservices. O'Reilly Media, Sebastopol
21. Wolff E (2017) Microservices: flexible software architecture. Addison-Wesley, Boston
22. Chen L. Microservices: architecting for continuous delivery and DevOps. In: Proceedings of the 2018 IEEE international conference on software architecture (ICSA); 2018; pp. 39–397.
23. Containerization explained. https://www.ibm.com/topics/containerization. Accessed 22 June 2023.
24. Turnbull J. The docker book: containerization is the new virtualization. James Turnbull. 2014
25 Cook J (2017) Docker hub. In: Cook J (ed) Docker for data science: building scalable and extensible data infrastructure around the Jupyter notebook server. Apress, Berkeley, pp 103–118
26. Sohan SM, Maurer F, Anslow C, Robillard MP. A study of the effectiveness of usage examples in REST API documentation. In: Proceedings of the 2017 IEEE symposium on visual languages and human-centric computing (VL/HCC). 2017; pp. 53–61.
27. Gokhale S, Poosarla R, Tikar S, Gunjawate S, Hajare A, Deshpande S, Gupta S, Karve K. Creating helm charts to ease deployment of enterprise application and its related services in kubernetes. In: Proceedings of the 2021 international conference on computing, communication and green engineering (CCGE); 2021; pp. 1–5.
28. Nelson KE, Scherer MK, et al. *JPype*; Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States), 2020.
29. The OpenAPI Specification (3.1.0). https://www.openapis.org. Accessed on 25 September 2023
30. Bremser W (1978) Hose—a novel substructure code. Anal Chim Acta 103:355–365. https://doi.org/10.1016/S0003-2670(01)83100-7
31. Ertl P, Roggo S, Schuffenhauer A (2008) Natural product-likeness score and its application for prioritization of compound libraries. J Chem Inf Model 48:68–74. https://doi.org/10.1021/ci700286x
32. DjoumbouFeunang Y, Eisner R, Knox C, Chepelev L, Hastings J, Owen G, Fahy E, Steinbeck C, Subramanian S, Bolton E et al (2016) ClassyFire: automated chemical classification with a comprehensive, computable taxonomy. J Cheminform 8:61. https://doi.org/10.1186/s13321-016-0174-y
33. Bento AP, Hersey A, Félix E, Landrum G, Gaulton A, Atkinson F, Bellis LJ, De Veij M, Leach AR (2020) An open source chemical structure curation pipeline using RDKit. J Cheminform 12:51. https://doi.org/10.1186/s13321-020-00456-1
34 Weininger D (1988) SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. J Chem Inf Comput Sci 28:31–36. https://doi.org/10.1021/ci00057a005
35. Heller SR, McNaught A, Pletnev I, Stein S, Tchekhovskoi D (2015) InChI, the IUPAC international chemical identifier. J Cheminform 7:23. https://doi.org/10.1186/s13321-015-0068-4
36 Heller SR, McNaught AD (2009) The IUPAC international chemical identifier (InChI). Chem Int Newsmag IUPAC 31:7–9. https://doi.org/10.1515/ci.2009.31.1.7
37 Weininger D, Weininger A, Weininger JL (1989) SMILES. 2. Algorithm for generation of unique SMILES notation. J Chem Inf Comput Sci 29:97–101. https://doi.org/10.1021/ci00062a008
38. Chemaxon Extended SMILES and SMARTS–CXSMILES and CXSMARTS. https://docs.chemaxon.com/display/docs/chemaxon-extended-smiles-and-smarts-cxsmiles-and-cxsmarts.md. Accessed 22 June 2023.
39 Krenn M, Häse F, Nigam A, Friederich P, Aspuru-Guzik A (2020) Self-referencing embedded strings (SELFIES): a 100% robust molecular string representation. Mach Learn Sci Technol 1:045024. https://doi.org/10.1088/2632-2153/aba947
40. Krenn M, Ai Q, Barthel S, Carson N, Frei A, Frey NC, Friederich P, Gaudin T, Gayle AA, Jablonka KM et al (2022) SELFIES and the future of molecular string representations. Patterns Prejud 3:100588. https://doi.org/10.1016/j.patter.2022.100588
41. Rajan K, Zielesny A, Steinbeck C (2021) STOUT: SMILES to IUPAC names using neural machine translation. J Cheminform 13:34. https://doi.org/10.1186/s13321-021-00512-4
42. Cahn RS, Ingold C, Prelog V (1966) Specification of molecular chirality. Angew Chem Int Ed Engl 5:385–415. https://doi.org/10.1002/anie.196603851
43 Rego N, Koes D (2015) 3Dmol.js: molecular visualization with WebGL. Bioinformatics 31:1322–1324. https://doi.org/10.1093/bioinformatics/btu829
44. Rajan K, Brinkhaus HO, Sorokina M, Zielesny A, Steinbeck C (2021) DECIMER-segmentation: automated extraction of chemical structure depictions from scientific literature. J Cheminform 13:20. https://doi.org/10.1186/s13321-021-00496-1
45 Rajan K, Brinkhaus HO, Isabel Agea M, Zielesny A, Steinbeck C (2023) DECIMER.ai—an open platform for automated optical chemical structure identification, segmentation and recognition in scientific publications. ChemRxiv. https://doi.org/10.26434/chemrxiv-2023-xhcx9
46. McKay BD, Yirik MA, Steinbeck C (2022) Surge: a fast open-source chemical graph generator. J Cheminform 14:24. https://doi.org/10.1186/s13321-022-00604-9
47. Schaub J, Zielesny A, Steinbeck C, Sorokina M (2020) Too sweet: cheminformatics for deglycosylation in natural products. J Cheminform 12:67. https://doi.org/10.1186/s13321-020-00467-y
48. Wilkinson MD, Dumontier M, Aalbersberg IJJ, Appleton G, Axton M, Baak A, Blomberg N, Boiten J-W, da Silva Santos LB, Bourne PE et al (2016) The FAIR guiding principles for scientific data management and stewardship. Sci Data 3:160018. https://doi.org/10.1038/sdata.2016.18
49. Hoyt CT, Zdrazil B, Guha R, Jeliazkova N, Martinez-Mayorga K, Nittinger E (2023) Improving reproducibility and reusability in the journal of cheminformatics. J Cheminform 15:62. https://doi.org/10.1186/s13321-023-00730-y
50. Prometheus Overview. https://prometheus.io/docs/introduction/overview/. Accessed 23 June 2023.
51 Chakraborty M, Kundan AP (2021) Grafana. In: Chakraborty M, Kundan AP (eds) Monitoring cloud-native applications: lead agile operations confidently using open source software. Apress, Berkeley, pp 187–240

52 Chandrasekara C, Herath P (2021) Introduction to GitHub Actions. In: Chandrasekara C, Herath P (eds) Hands-on GitHub actions: implement CI/CD with GitHub action workflows for your applications. Apress, Berkeley, pp 1–8
53. Gaulton A, Hersey A, Nowotka M, Bento AP, Chambers J, Mendez D, Mutowo P, Atkinson F, Bellis LJ, Cibrián-Uhalte E et al (2017) The ChEMBL database in 2017. Nucleic Acids Res 45:D945–D954. https://doi.org/10.1093/nar/gkw1074
54. Taneja S, Gupta PR (2014) Python as a tool for web server application development. JIMS8I Int J Inf 2:77–83
55. Hanson RM, Musacchio S, Mayfield JW, Vainio MJ, Yerin A, Redkin D (2018) Algorithmic analysis of Cahn–Ingold–Prelog rules of stereochemistry: proposals for revised rules and a guide for machine implementation. J Chem Inf Model 58:1755–1765. https://doi.org/10.1021/acs.jcim.8b00324
56. John M (2018) Centres: perception and labelling of stereogenic centres in chemical structures *(Version 10) [Computer software]*. Github, San Francisco
57. Herráez A (2006) Biomolecules in the computer: Jmol to the rescue. Biochem Mol Biol Educ 34:255–261. https://doi.org/10.1002/bmb.2006.494034042644
58. Hanson RM, Prilusky J, Renjian Z, Nakane T, Sussman JL (2013) JSmol and the next-generation web-based representation of 3D molecular structure as applied to Proteopedia. Isr J Chem 53:207–216. https://doi.org/10.1002/ijch.201300024
59. PubChem Testosterone. https://pubchem.ncbi.nlm.nih.gov/compound/6013. Accessed 23 June 2023.
60. Dai G, Sun J, Peng X, Shen Q, Wu C, Sun Z, Sui H, Ren X, Zhang Y, Bian X (2023) Astellolides R-W, drimane-type sesquiterpenoids from an *Aspergillus Parasiticus* strain associated with an isopod. J Nat Prod. https://doi.org/10.1021/acs.jnatprod.3c00215
61. Senart T. Vegeta: HTTP load testing tool and library: it's over 9000. Github.

Publisher's Note