The Journal of Open Source Software

optimade-python-tools: a Python library for implementing and consuming materials data via OPTIMADE APIs

Matthew L. Evans[1, 2], **Casper Andersen**[3], **Shyam Dwaraknath**[4], **Markus Scheidgen**[5], **Ádám Fekete**[1, 6, 8], **and Donald Winston**[4, 7]

1 Institut de la Matière Condensée et des Nanosciences, Université catholique de Louvain, Chemin des Étoiles 8, Louvain-la-Neuve 1348, Belgium **2** Theory of Condensed Matter Group, Cavendish Laboratory, University of Cambridge, J. J. Thomson Avenue, Cambridge, CB3 0HE, U.K. **3** EPFL **4** LLBL **5** FHI **6** Namur **7** Polyneme LLC **8** KCL

DOI: 10.21105/joss.0XXXX

Software

- Review
- Repository
- Archive

Editor: Editor Name

Submitted: 01 January XXXX
Published: 01 January XXXX

Summary

In recent decades, improvements in algorithms, hardware and theory have enabled crystalline materials to be studied at the atomistic level with great accuracy and speed. To enable dissemination, reproducibility, and reuse, many digital crystal structure databases have been created and curated, ready for comparison with existing infrastructure storing structural characterizations of real crystals. These databases are often made available with bespoke, application programming interfaces (APIs) to allow for automated, and often open, access to the underlying data. Such esoteric APIs incur maintenance and usability costs upon both the data providers and consumers, neither of whom may necessarily be software specialists.

The OPTIMADE API specification (C. Andersen et al., 2020; C. W. Andersen et al., 2021), released in July 2020, aimed to reduce these costs by designing a common API for use across a consortium of collaborating materials databases. Whilst based on the robust JSON:API standard (*The JSON*, n.d.), the OPTIMADE API specification presents several domain-specific features and requirements that can be tricky to implement for non-specialist teams. The package presented here, `optimade-python-tools`, provides a modular reference server implementation and a set of associated tools to accelerate the development process for data providers, toolmakers and end-users themselves.

Statement of need

In order to accommodate existing materials database APIs, the OPTIMADE specification allows for flexibility in the specific data served but enforces a simple, but domain-specific, filter language on well-defined resources. This flexibility could be daunting to database implementers and maintainers and could act to increase the activation barrier to hosting an API. `optimade-python-tools` aims to catalyse the creation of APIs from existing and new data sources by providing a configurable and modular reference server implementation for hosting materials data in an OPTIMADE-compliant way. The package leverages the modern Python libraries pydantic (*Pydantic*, n.d.) and FastAPI (*FastAPI*, n.d.) to specify the data models and API routes defined in the OPTIMADE specification, additionally providing a schemas following the OpenAPI format (*The OpenAPI Specification*, n.d.). Two storage back-ends are supported out of the box, with full filter support for databases that employ the popular MongoDB (*MongoDB*, n.d.) or Elasticsearch (*Elasticsearch*, n.d.) frameworks.

The Journal of Open Source Software

Functionality

The modular functionality of `optimade-python-tools` can be broken down by the different stages of a user query to the reference server. Consider the following query URL:

```
optimade.example.org/v1/structures?filter=chemical_formula_anonymous="ABC"
```

This query should match any crystal structures in the database with a composition that consists of any three elements in a 1:1:1 ratio. The "anatomy" of this query is displayed in Figure 1.

1. After routing the query to the appropriate `/structures/` endpoint adhering to `v1` of the specification, the filter string `chemical_formula_anonymous="ABC"` is tokenized and parsed into an abstract tree by a `FilterParser` object using the Lark parsing library (**Lark?**) against the Extended Backus-Naur Form (EBNF) grammar defined by the specification.
2. The abstract tree is then transformed by a `FilterTransformer` object into a database query specific to the configured back-end for the server. This transformation can include aliasing and custom transformations such that the underlying database format can be accommodates.
3. The results from the database query are then deserialized by `EntryResourceMapper` objects into the OPTIMADE-defined data models and then re-serialized into JSON before being served to the user over HTTP.

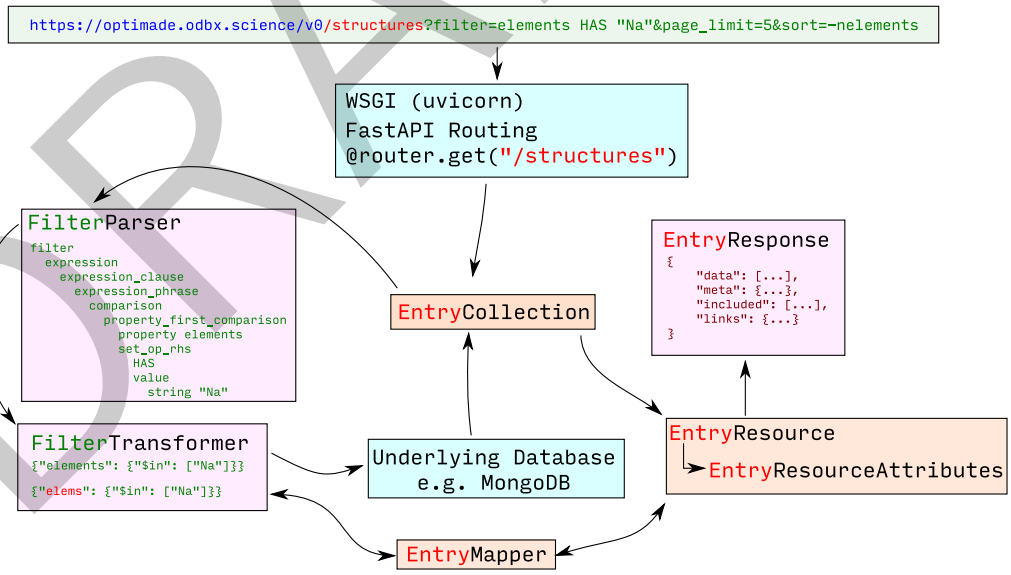

Figure 1: Anatomy of an OPTIMADE query handled by the library.

Beyond this query functionality, the package also provides:

- A fuzzy implementation validator that performs HTTP queries against remote OPTIMADE APIs, with test queries and expected responses generated dynamically based on the data served at the introspective `/info/` endpoints of the API implementation.
- Entry "adapters" that can convert between OPTIMADE-compliant entries and the data models of the popular Python libraries `pymatgen` (Ong et al., 2013) and `ase` (the Atomic Simulation Environment) (Larsen et al., 2017).

Use cases

The package is currently used in production by three major data providers for atomistic data:

- The Materials Project uses `optimade-python-tools` alongside their existing API (**MAPI?**) and MongoDB database, providing access to highly-curated density-functional theory calculations across all known inorganic materials. `optimade-python-tools` handles filter parsing, database query generation and response validation by running the reference server implementation with minimal configuration.
- The NoMaD Repository and Archive integrates the `optimade-python-tools` server in an existing web app and uses the Elasticsearch implementation of the filtering module to allow access to 100M(?) published first-principles calculations submitted by users.
- Materials Cloud uses `optimade-python-tools` to provide domain-specific API access to published calculations that were created with AiiDa (**AiiDa?**) and archived on their system. In this case, each individual archive entry has its own database and separate API. The classes within `optimade-python-tools` have been extended to make use of AiiDa and its underlying PostgreSQL (**PostgreSQL?**) storage engine.

Acknowledgements

M.E. would like to acknowledge the EPSRC Centre for Doctoral Training in Computational Methods for Materials Science for funding under grant number EP/L015552/1 and support from the European Union's Horizon 2020 research and innovation program under the European Union's Grant agreement No. 951786 (NOMAD CoE).

Andersen, C., Armiento, R., Blokhin, E., Conduit, G., Dwaraknath, S., Evans, M. L., Fekete, Á., Gopakumar, A., Gražulis, S., Merkys, A., Mohamed, F., Oses, C., Pizzi, G., Rignanese, G.-M., Scheidgen, M., Talirz, L., Toher, C., & Winston, D. (2020). *The OPTIMADE specification* (Version 1.0) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.4195051

Andersen, C. W., Armiento, R., Blokhin, E., Conduit, G. J., Dwaraknath, S., Evans, M. L., Fekete, Á., Gopakumar, A., Gražulis, S., Merkys, A., Mohamed, F., Oses, C., Pizzi, G., Rignanese, G.-M., Scheidgen, M., Talirz, L., Toher, C., Winston, D., Aversa, R., … Yang, X. (2021). *OPTIMADE: An API for exchanging materials data*. http://arxiv.org/abs/2103.02068

Elasticsearch (Version 6.4). (n.d.). https://www.elastic.co

FastAPI (Version 0.65.1). (n.d.). https://github.com/tiangolo/fastapi

Larsen, A. H., Mortensen, J. J., Blomqvist, J., Castelli, I. E., Christensen, R., Dułak, M., Friis, J., Groves, M. N., Hammer, B., Hargus, C., Hermes, E. D., Jennings, P. C., Jensen, P. B., Kermode, J., Kitchin, J. R., Kolsbjerg, E. L., Kubal, J., Kaasbjerg, K., Lysgaard, S., … Jacobsen, K. W. (2017). The atomic simulation environment—a Python library for working with atoms. *J. Phys.: Condens. Matter*, *29*(27), 273002. https://doi.org/10.1088/1361-648x/aa680e

MongoDB (Version 4.4). (n.d.). https://www.mongodb.com

Ong, S. P., Richards, W. D., Jain, A., Hautier, G., Kocher, M., Cholia, S., Gunter, D., Chevrier, V. L., Persson, K. A., & Ceder, G. (2013). Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, *68*, 314–319. https://doi.org/10.1016/j.commatsci.2012.10.028

Pydantic (Version 1.8.2). (n.d.). https://github.com/samuelcolvin/pydantic

The JSON:API specification (Version 1.0). (n.d.). https://jsonapi.org/format/1.0/

The OpenAPI specification (Version 3.1). (n.d.). https://spec.openapis.org/oas/v3.1.0