MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Design and Optimization of a Quantum Materials Database"

verfasst von / submitted by

Gudrun Pötzelberger, MSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 910
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Computational Science UG2002
Betreut von / Supervisor:	Univ.-Prof. Dr.techn. Cesare Franchini

Abstract

This thesis introduces the GW-Database, a computational materials database specializing in GW data. Materials databases are widely used in the field of materials science, but most open-access materials databases focus exclusively on density functional theory (DFT) data. However, it is well-known that DFT systematically underestimates bandgaps. The GW approximation is a method that gives more accurate results, but is more computationally expensive. The GW-Database provides access to accurate GW data. The GW-Database was improved from its original state to add a number of vital features that allow it to fulfill its potential. Among other improvements, data about the bandgaps was added, functional bandstructure plots were created, and the presentation of the data was restructured into a more logical layout. A detailed description of the frontend, backend, and database comprising the GW-Database is provided in this thesis.

Zusammenfassung

Diese Arbeit stellt die GW-Database vor, eine Datenbank für computergestützte Materialdaten, die auf GW-Daten spezialisiert ist. Materialdatenbanken finden in der Materialwissenschaft breite Anwendung, jedoch konzentrieren sich die meisten frei zugänglichen Datenbanken ausschließlich auf Daten die mit der Dichtefunktionaltheorie (DFT) ausgerechnet wurden. Es ist allerdings bekannt, dass DFT systematisch Bandlücken unterschätzt. Die GW-Approximation ist eine Methode die genauere Ergebnisse liefert, aber auch mehr Rechenleistung erfordert. Die GW-Database stellt Zugriff auf genaue GW Daten zur Verfügung. Die GW-Database wurde von ihrem ursprünglichen Zustand verbessert, indem eine Reihe wichtiger Eigenschaften hinzugefügt wurden um ihr Potential voll auszuschöpfen. Unter anderem wurden Daten zu den Bandlücken hinzugefügt, korrekt funktionierende Bandstrukturdiagramme wurden erstellt, und die Darstellung der daten wurde in eine logischere Struktur umgestaltet. Eine detaillierte Beschreibung des Frontend, Backends und der Datenbank, die zusammen die GW-Database ausmachen, wird in dieser Arbeit zur Verfügung gestellt.

Contents

Chapter 1

Introduction

1.1 Materials Databases

A materials database is an organized collection of information about different materials and their properties. Materials databases play a crucial role in the field of Materials Science, because they enable researchers to gain access to vast amounts of data collected by their peers, while also providing them with a platform to share their own findings. There exists a huge amount of materials and materials data, making the task of organizing and analyzing it an exceedingly challenging task. Having open databases available enables researchers to effectively navigate this large amount of information, which ultimately leads to more scientific discoveries and innovations in the field.[1]

The significance of materials databases encompasses both experimental and computational materials data. Obtaining experimental data often requires complex and time-consuming experiments, which generate a lot of valuable information. Predictive computer simulations based on first principles approaches offer a cheaper alternative to gain useful data, which leads to a transparent and sound interpretation of materials properties. Simulation data can also be used to decide which materials merit a more costly experimental examination. Sharing this data enables other researchers to validate findings, build upon existing work, and conduct meta-analyses, fostering a collaborative environment where scientific knowledge can grow collectively.

Another important use case for materials databases is the training of machine learning models. Machine learning algorithms can identify patterns, correlations, and hidden trends in the data, leading to the discovery of novel materials with specific desired properties. They can also compute materials properties more efficiently than first principles methods. However, training effective machine learning models requires vast amounts of data. Open-access databases, featuring extensive data sets in a standardized format, play a crucial role in facilitating the training process.

There currently exist several open-access databases for computational materials data, some of the most widely used of which are listed below.

1. **The Materials Project**[2]
   The Materials Project is an open-access database of materials simulation data and a major part of the Materials Genome Initiative[3]. Density functional theory[4] (DFT) is used to calculate materials properties[5,6], which can then be freely accessed by researchers.
2. **The AFLOW (Automatic Flow) database**[7] The AFLOW database incorporates DFT data from a variety of sources, including computational results from AFLOW calculations, data from other databases, and scientific literature. AFLOW is a computational framework that automates DFT calculations.
3. **NoMaD (Novel Materials Discovery)**[8] Researchers can make their raw data public by uploading it to the NoMaD repository. It also contains the data of AFLOW, OQMD, and The Materials Project. The NoMaD archive provides the same data as the NoMaD repository, but in a normalized format. NoMaD accepts data calculated by different kinds of simulation codes.
4. **OQMD (Open Quantum Materials Database)**[9] The OQMD focuses on providing high-quality comparable data, calculated using DFT. In order to ensure comparability between materials, all DFT calculations are performed with consistent parameters, such as the plane-wave cutoff or the k-point density.[10]

Even though nowadays all these and more open-access *ab initio* simulations databases are available, most of them focus predominately or entirely on DFT data.[6,7,10] DFT is still considered the state of the art in the field of *ab-initio* materials simulations. However, DFT has some shortcomings in regards to accuracy for certain properties.[11] In particular, it systematically underestimates bandgaps.[12]

The GW approximation[13] is a beyond-DFT method that fixes this shortcoming by using a more rigorous treatment of electron-electron interactions through a direct estimation of the so-called self-energy. However, it is also more computationally expensive.[14] GW simulation results are a useful reference for materials researchers, due to their higher accuracy compared to DFT calculations. Using more accurate reference data from GW calculations allows researchers to benchmark and validate the performance of DFT methods, identify discrepancies and limitations in their predictions, and make more informed decisions in selecting appropriate methods for their calculations.

1.2 Thesis Goal

The goal of this thesis is to develop a small, open-access GW database to allow researchers in the Quantum Materials Modelling Group headed by Cesare Franchini at the University of Vienna, as well as scientists in other research groups, to share their GW data. While DFT databases are abundant, the scarcity of open-access GW databases presents a unique opportunity to create a valuable resource. Having a dedicated GW database offers a centralized platform to store GW results and share them with the wider materials science community. This database's novelty lies in its focus on GW data, which will allow researchers to access highly accurate reference data at the beyond-DFT level, validate their DFT predictions, and enhance the understanding of critical electronic properties. By facilitating the exchange of GW data and promoting collaboration, this GW database has the chance to advance materials research and accelerate scientific discoveries within the group and beyond.

To achieve this goal, the present thesis will involve using a preexisting database that was designed for the Quantum Materials Modelling Group to publish beyond-DFT data. The database was originally created by Karl Prikopa in 2016 for this purpose, but presently lacks certain features, as well as documentation. In my investigation, I will examine the workflow and data-passing mechanisms employed by this GW-Database, while also exploring the workflow used in VASP to acquire the results to be incorporated into the database. Furthermore, an essential aspect of this research will focus on optimizing the GW-Database by addressing the identified shortcomings and introducing the missing features, thereby enhancing its functionality and utility as a robust platform for GW data storage and analysis.

Chapter 2

First-Principles Calculations

Ab-initio simulations allow the prediction of the properties of a material from first principles (i.e. without using any phenomenological parameters), and are widely used in materials science. They have helped increase understanding of material properties, as well as accelerated the discovery of novel materials through predictive modeling.

The fundamental objective of ab-initio methods is to use approximations to solve the many-body Schrödinger equation, which offers a complete description of the quantum state of a system. However, solving this equation directly is exceedingly difficult due to its complexity and the exponential increase in computational demands as the number of interacting particles grows. For this reason, approximations are necessary.

2.1 Density Functional Theory

The most popular approach in current computational materials science is DFT. In DFT, all properties of the quantum-mechanical ground state are calculated from the electron density. This allows the modeling of systems with more electrons, since the electron density depends on only three spatial variables, while the wavefunction of an N-electron system is a function of $3N$ spatial variables.[15] However, standard DFT describes only ground-state properties and not electronic excitations or quasi-particle energies.[11]

When using the DFT approximation, the so called Kohn-Sham system is mathematically constructed. In this system, the electrons are non-interacting particles, but it still has the same ground-state electron density as the original system. Instead of interactions between electrons, the non-interacting electrons are approximated to be moving in an effective potential. Since the total energy of the system is uniquely determined by the system's electron density[4], the energy of the Kohn-Sham system is the same as the energy of the original system. This results in N independent Kohn-Sham equations, one for each electron. Solving the Kohn-Sham system's equations, similar to the Schrödinger equation of a real system, yields the Kohn-Sham orbitals.[16] The Kohn-Sham equation for the ith electron is

$$\left(-\frac{\hbar}{2m_e}\nabla^2 + V_{en}(\mathbf{r}) + V_H + V_{xc}(\mathbf{r})\right)\phi_i(\mathbf{r}) = \epsilon_i^{KS}\phi_i(\mathbf{r}) \tag{2.1}$$

, where V_{en} is the nucleus-electron potential, V_H is the Hartree potential (i.e. the classical electrostatic potential resulting from all electrons[17]), V_{xc} is the exchange-correlation potential, ϕ_i is the Kohn-Sham orbital, and ϵ_i is the Kohn-Sham eigenvalue.

DFT has some well-known shortcomings[11], most prominently the systematic underestimation of bandgaps[12,18]. This is a significant drawback, since the bandgap, and the band energies more broadly, define many materials properties and are of great interest in modern materials science, for example in the design and optimization of photovoltaic devices.

2.2 GW Approximation

One method that solves DFT's problem of underestimating bandgaps is the so-called GW approximation[19,13], which corrects the self-energy term of the correlation. It is a beyond-DFT method that provides a better description than DFT of band structure and dielectric properties, though at a significantly higher computational cost.[14]

'G' stands for the one-body Green's function, which describes the system's response to perturbations, and 'W' stands for the screened Coulomb potential, taking into account that the Coulomb interaction between two electrons is partially screened by the presence of other electrons in the material. G and W together describe the electron self-energy $\Sigma = iGW$, which is a key property in the GW approximation[17] and contains the effects of exchange and correlations.[13] To run a GW calculation, Kohn-Sham orbitals of a fully converged DFT calculation are typically used as an initial guess for the unperturbed system.

The central equation for a GW calculation is

$$\left(-\frac{1}{2}\Delta + V_{en}(\mathbf{r}) + V_H(\mathbf{r})\right)\psi_{n\mathbf{k}}(\mathbf{r}) + \int \Sigma(\mathbf{r},\mathbf{r}',E_{n\mathbf{k}})\psi_{n\mathbf{k}}(\mathbf{r}')d\mathbf{r}' = E_{n\mathbf{k}}\psi_{n\mathbf{k}}(\mathbf{r}) \tag{2.2}$$

, where $\mathbf{k}$ is a Bloch wave vector and n is a band index, V_{en} is the nucleus-electron potential, V_H is the Hartree potential, Σ is the many-body self-energy, and $E_{n\mathbf{k}}$ are the quasiparticle energies with the orbitals $\psi_{n\mathbf{k}}(\mathbf{r})$.[13,20,21]

The difference between the Kohn-Sham eigenvalues of the initial DFT calculation and the GW-corrected eigenvalues is called the quasi-particle shift. It represents the correction that GW makes to the DFT energy eigenvalues. The shift is not completely uniform across the bandstructure, and can change slightly depending on the *k*-point at which it is calculated. It is usually calculated for the valence band, as that band tends to experience the greatest shift, in most cases downward. Since the bandgap is the property of highest interest, the shift is sometimes additionally calculated for the conduction band, as the combination between the valence band shift and the conduction band shift represents the overall correction of the bandgap.[22]

Both G and W can be iterated to achieve self-consistency. However, a computationally cheaper approach is the G_0W_0[23] method, also known as single-shot GW.[24] When using this method, only one GW step is performed, meaning the Kohn-Sham orbitals are not changed from the initial guess and the self-energy and the quasiparticle energies are not iteratively updated.

In the conventional approach, the DFT calculation for the initial guess as well as the G_0W_0 calculation itself need to be repeated several times while step-wise increasing computational parameters such as the energy cutoff, and the number of *k*-points and (empty) bands. As soon as the output no longer changes for higher parameters, the parameters are considered to be converged and can be used for the actual calculation, as well as any future calculations involving this system.

Finding the converged parameters in this way is time-consuming work, in addition to not being mathematically rigorous in the sense that it is not guaranteed that using these converged parameters leads to accurate results. One way to improve this situation for individual properties is to use the basis set extrapolation method[25,22] (BS-EX) instead. With this method, four GW calculations are performed and their results are combined to perform a linear extrapolation to an infinitely large basis set. For details on this method, please refer to[22]. This method only results in knowledge about individual properties, such as bandgaps, rather than knowledge about the system as a whole. For instance, it cannot be used to extract information about the dielectric function. The extrapolation of the corrected bandgaps is not done in VASP automatically, but is a separate calculation done by the researcher.

Despite not being self-consistent, results obtained with the G_0W_0 approximation have been found to be in good agreement with experimental data, especially for bulk systems.[26,27,28] As one might expect, G_0W_0 is more dependent on the starting point (usually calculated with DFT) than self-consistent GW, which is why it is important to specify the used DFT functional along with other computational factors to make the calculations reproducible.

In this project, the G_0W_0 method was used to calculate the results that can be found in the database, and the employed DFT functional can be found in the downloadable OUTCAR file.

Chapter 3

Programming Methods

3.1 Database Management Systems

To be able to store and access large amounts of data efficiently, database management systems (DBMS) are needed. A DBMS consists of a structured collection of data that is stored in such a way as to make it easy to retrieve and manipulate (the database), and the interface over which this retrieval and manipulation is done. The goal of a DBMS is to manage data in a way that is convenient and efficient, satisfying such requirement as data consistency and easy querying.[29]

There are different kinds of DBMS. One of the most popular types is called a relational[30] DBMS, which represents data in the form of connected tables. Many relational DBMS use Structured Query Language (SQL) as a language to retrieve, insert, update, and delete data from the database. In contrast, non-relational DBMS which use other query languages are called noSQL systems. The structure of a relational database can be represented by an Entity-Relationship diagram, also called ER-diagram. In this project, the open-source relational DBMS MariaDB[31] is used. It is an open-source fork of the MySQL[32] DBMS and was designed to be fully compatible with it, so it is possible to run this project with either MySQL or MariaDB.

In a relational DBMS, each table has a primary key, consisting of one or more columns of the table. These columns that make up the key uniquely identify each row of data, meaning each row of the table must have different data in the key columns. Often, an ID-column is added to act as the primary key, as this allows for easier data handling.

To represent the ways that the tables are interconnected among each other, foreign keys are used. The way they are used depends on the specific relations between the tables (one-to-one relation, a one-to-many relation, or a many-to-many relation), but their basic function is that they allow a table to reference the primary key of another table. In general, foreign keys are used in order to create connections between tables. Foreign keys are not strictly necessary, but they do enforce referential integrity, as without foreign keys, it would be possible to reference a row of data that does not exist.[29]

SQL allows for complex SELECT queries which are used to retrieve data from the database. SQL, being a declarative language, only requires users to specify the data they require from a database without having to explicitly define the retrieval process. The DBMS internally optimizes the query to ensure efficient data retrieval, utilizing various optimization techniques like index usage, query plan optimization, and caching mechanisms. Of particular note is the ability to JOIN tables, that is, to combine data from different tables based on values in specified columns. In this way, related data from different tables can be connected together in a query.

Furthermore, SQL permits users to save the results of a SELECT query as a virtual table, known as a view. Views function like regular tables but hold references to the underlying tables, ensuring any changes made to the source data are reflected in the view.

In addition to the SELECT command for data retrieval, SQL also supports INSERT, UPDATE, and DELETE operations. Users can combine the SELECT statement with these commands to specify precisely which data they want to apply the changes to, enabling efficient data manipulation within the database.

3.2 Model-View-Controller

The Model-View-Controller (MVC) pattern is a software design pattern that is extensively used in applications with user interfaces. In this pattern, the software is divided into three parts: The model representing the data, the view which presents the data to the user, and the controller which handles user input and acts as intermediary between model and view. The controller is responsible for application flow and passing on user requests to change the data to the model.[33]

The advantage of this pattern is that its modular nature enables the development of flexible and extensible software, since each of the three components can be changed or replaced without having to alter the two other components. This improves maintainability as well as reusability, as each of the three components can be switched out in a modular way and reused somewhere else.[33]

Furthermore, this pattern supports multiple views connected to the same model, meaning that the same data can be presented across different devices, such as phones, smartwatches, and desktops. Typically, each view has a corresponding controller which defines the response to user inputs. Changing the active controller of a view changes the way the application reacts to user inputs without necessitating any change to model or view.[33]

3.3 API

An API (application programming interface) is a software component that allows communication and data exchange between different software systems. One possible use case of an API is to facilitate communication between frontend and backend of an application.

In general, an API allows different software systems to work together by defining how requests and responses should be structured. In this way, only the part of the software that does the actual communicating needs to adapt to the data structures and conventions of its communication partner, while the rest of the software can have a completely different structure.

The standardized way of data exchange that is made possible by APIs helps to simplify and abstract data exchanges without sacrificing internal complexity of either communication partner. If a software provides an API to be communicated with, the only knowledge needed to communicate with it is knowledge about the API itself, not about the underlying system or how it works internally.

Chapter 4

The GW-Database

The database that is the subject of this thesis (hereafter called ‘GW-Database’) was first created by Karl Prikopa for the Quantum Materials Modelling Group at the University of Vienna. Its purpose is to store and provide accurate data on the bandgaps of numerous insulating and semiconducting materials. The highly accurate ab-initio GW approximation is used to produce the data for the GW-Database.

The converged input parameters of the calculations are also provided for each material in the database, notably the number of bands (NBANDS) and the number of k-points (KPOINTS). Obtaining these converged parameters can take much time and effort, so their inclusion in the database may prove useful to anyone who wishes to recreate the calculations.

4.1 Materials in the Database

The materials in the database were chosen so as to cover a wide range of different types of semiconductors and insulators, including monoatomic, binary, and ternary compounds, with bandgaps ranging from 0.1 eV to 20 eV at Γ. It includes benchmark materials such as Si or C in different phases.

Figure 4.1 shows the frequency of elements in all distinct materials that are available in the GW-Database. As the figure shows, a wide range of elements are represented, with a special focus on elements that are frequent in semiconductors and insulators. There are currently calculation results for 124 different materials stored in the GW-Database.

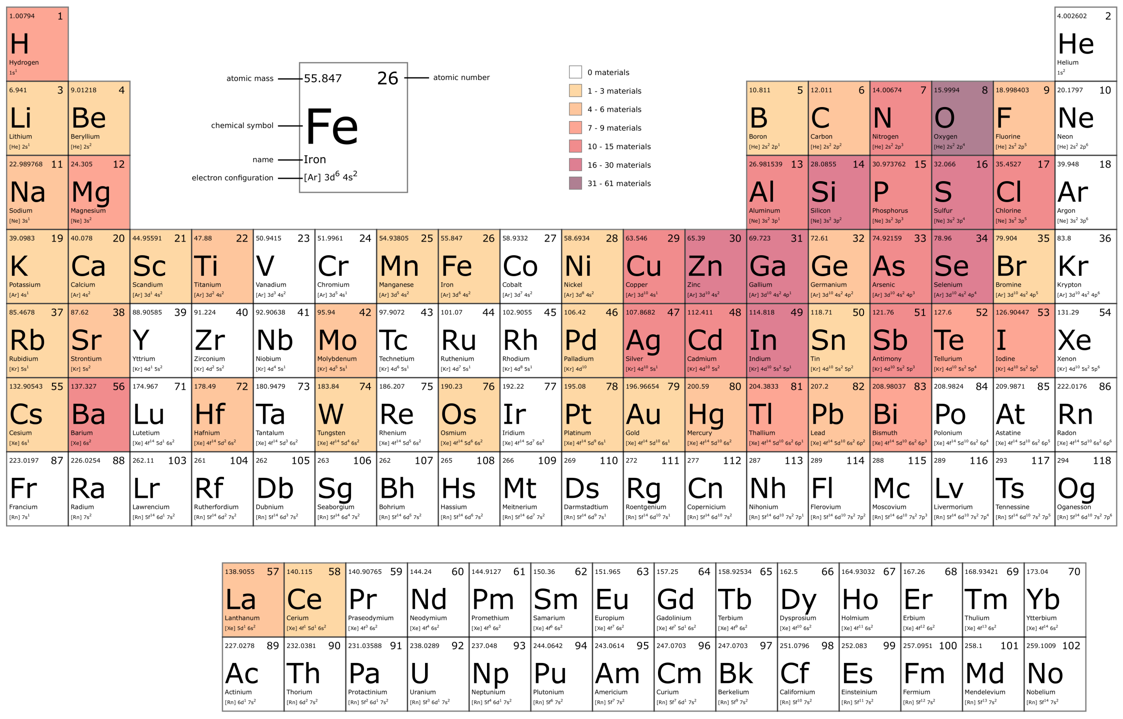


Figure 4.1: The frequency of elements in all distinct materials that are currently available in the GW-Database.

4.2 VASP calculations

The data in the GW-Database was obtained from *ab-initio* electronic structure calculations performed by the Vienna Ab-Initio Simulation Package[34,35] (VASP), a popular software tool in computational materials science. VASP uses a plane-wave basis set and periodic boundary conditions to simulate the quantum mechanical properties of materials.

A VASP calculation requires at least the following four kinds of input files: INCAR, POSCAR, KPOINTS, and POTCAR. The INCAR file is used to specify the details of how the computation should be performed, such as the energy cutoff, the number of bands, which output should be produced, and generally which kind of simulation should be executed (e.g. DFT or GW). The POSCAR file contains the geometry of the system, such as lattice vectors and the coordinates and types of atoms. The KPOINTS file specifies the density and centering of the k-points mesh in reciprocal space, or in the case of bandstructure calculations, the high-symmetry k-points path in the first Brillouin zone along which the bandstructure should be calculated. Lastly, the POTCAR file contains the pseudopotentials for each kind of atom that is contained in the material. In this project, GW-optimized versions of POTCARs with Projector Augmented Wave[36] (PAW) potentials[37] for Perdew-Burke-Ernzerhof[38] (PBE) calculations were used. POTCAR files are not user-generated, but supplied by VASP. They are copyrighted and therefore are not saved in the GW-Database. However, information about which POTCAR file was used can be found in the output files of a calculation.

Depending on the type of calculation, different files are created as the output. The standard output files that are always produced are OUTCAR and vasprun.xml, which contain results of the calculation, such as forces, dielectric properties, and information about the calculation, as well as a summary of the used input parameters.[39]

The workflow of using VASP to obtain simulation results which are suitable to be uploaded into the GW-Database can be summarized as follows:

1. DFT Calculation
   In the first step, the ground state is calculated with a standard DFT calculation. This step provides the ground state Kohn-Sham orbitals of the system.

2. Bandstructure Calculation (Optional)
   In this optional step, the DFT bandstructure can be calculated starting from the output files of step 1, and using a KPOINTS file specifying the path through the Brillouin zone along which the bandstructure should be calculated[40]. If this step is left out, the bandstructure cannot be plotted.

3. DFT NBANDS
   Starting from the output files of step 1, the number of NBANDS is increased in the INCAR file, and the LOPTICS flag is set to true, which tells VASP to generate a WAVEDER file, which is a file containing the derivative of the wavefunction. The INCAR file is configured such that only one step of the simulation is executed, just to add the additional (empty) bands.

4. G_0W_0
   The WAVECAR and WAVEDER files that resulted from step 3 are used as input files, and the G0W0 algorithm is specified in the INCAR file. Then, the single-shot GW calculation is executed. One notable result of this step is the GW-bandgap, along with many other material properties.

Steps 1, 3, and 4 all need to be repeated multiple times for different input parameters, such as the energy cutoff or the number of k-points, in order to converge these parameters in parameter-space. Once a set of converged input parameters has been found, it can be used for all future calculation concerning the specific system. However, methods are being explored that avoid having to manually converge these input parameters, such as the basis set extrapolation described in Section 2.2.

For the data that is currently in the database, calculations were performed using PAW pseudopotentials and PBE exchange-correlation functionals, though it is of course possible to upload data calculated in different ways to the GW-Database. The exact potential that was used for each calculation can be discovered by checking its OUTCAR or vasprun.xml file.

The VASP calculations were executed on the Vienna Scientific Cluster[41] (VSC), which provides supercomputing resources to Austrian universities. In the workflow above, the G_0W_0 calculation in step 4 usually takes up over 95% of the overall computing time.

4.3 Description of the GW-Database

As the GW-Database was originally created without any documentation, it will be described here. This will not only give a useful overview over the project and its current state, but also make it easier to build upon it in the future. The goal is to allow readers to gain a basic understanding of the functionality and structure of the project, and to help future maintainers to avoid having to spend months familiarizing themselves with the source code.

In this project, PHP[42] (PHP: Hypertext Preprocessor) is used as a server-side language, and JavaScript[43] is used as a client-side language.

The GW-Database code consists of three components:

1. The MariaDB database, which stores the data itself.
2. The backend, written in PHP, which parses and processes the data, accesses the MariaDB Database to store or retrieve the data, and implements an API that can be queried for the data.
3. The frontend, written in HTML and JavaScript, which presents the data to the user.

The frontend communicates with the backend through a API with a well-defined response structure. The fact that this API is the only connection between backend and frontend means it would be easy to add another frontend with a different presentation of the data, such as a phone app, without having to change the existing code in any way. The new frontend could simply query the same API as the current frontend, and then process and show the data in a different way. This API can also be queried directly by the user to allow convenient automated data access.

The GW-Database is based on the Model-View-Controller Pattern (MVC) as described in Section 3. In practice, this allows the backend and the database to be developed independently of the frontend.

In terms of the previously listed components of the code, this implementation of the MVC pattern is structured in the following way, which is also shown in Figure 4.2: The model consists of the combination of the MariaDB database and the backend. The view consists of the frontend that is visible to the user, mostly written in HTML, but with some JavaScript parts such as the plots showing the data. Lastly, the controller is the JavaScript code which reacts to user inputs and makes API calls to the backend to fetch or alter the data of the model.

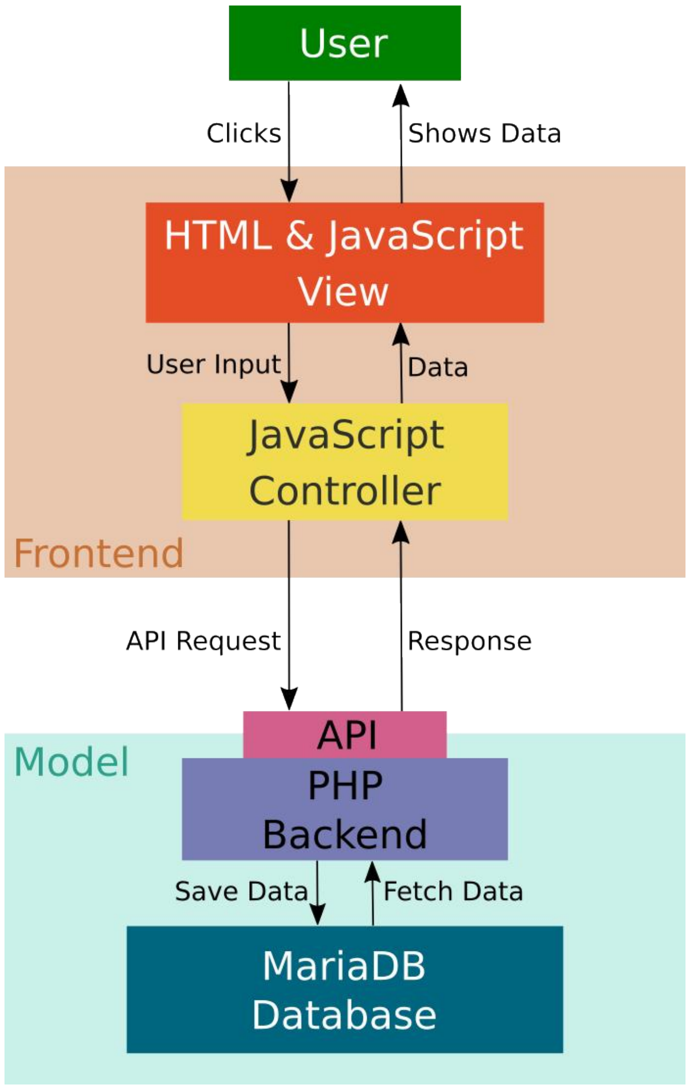


Figure 4.2: The structure of the GW-Database. The user interacts with the frontend, which consists of the view and the controller. Only the view is visible to the user, but the controller controls how the view responds to user input. The model, consisting of the database and the backend, provides access via the API. The controller can interface with the model by making API requests, which can cause the model to modify its data or send data in as response to the API call. In this way, the controller can get the data from the model and write it into the view so that the user can see it.

In the following sections, the three components of the GW-Database—database, backend, and frontend—will be described in detail.

4.3.1 Database

All available information except for the VASP files is stored in the database. For this project, a MariaDB database is used, as described in Section 3. Figure 4.3 shows the most important parts of the database in the form of an ER-diagram. As the database was provided without documentation, this ER-diagram was reverse-engineered from the database itself, rather than being the original model that was used to construct the database. For space and clarity, the diagram does not include all existing entities and attributes, but focuses on the most important ones.

The central tables in the database are `materials`, `poscar`, `incar`, and several tables holding the results-data gained from the OUTCAR and vasprun.xml files: `outcar_mod_bandstructure`, `outcar_mod_computation`, `outcar_mod_general`, and `outcar_mod_imdt`. These tables are connected to each other through the `materials_list` table.

As the name suggests, the `poscar` table contains the information that can be parsed from the POSCAR file, such as the lattice constants and angles, and the volume of the cell. Each entry is identified by its unique `poscar_id`. The `incar` table contains the data that would be written in INCAR file, such as the number of bands and the energy cutoff. This data is parsed from the OUTCAR file, as the INCAR file itself is not uploaded. The number of k-points in x, y and z direction is also stored in the `incar` table. For the optional keywords, the table `incar_lines` is used to store the keywords and its value as key-value pairs. Each row of the `incar` table is identified by a unique ID, whereas for the `incar_lines` table several rows can have the same `incar_id`, since one INCAR file will usually have more than one key-value pair. The primary key for the `incar_lines` table is a composite key made up of the `incar_id`, the number of the line on which the keyword is written in the INCAR file, and the type of calculation (e.g. DFT or G0W0).

The tables `outcar_mod_bandstructure`, `outcar_mod_computation`, `outcar_mod_general`, and `outcar_mod_imdt` all have the column `outcar_id` as a primary key. Rows in different tables with the same ID in the `outcar_id` column contain data belonging to the same overall calculation, though not necessarily to the same OUTCAR or vasprun.xml file. The table `outcar_mod_bandstructure` contains mainly data parsed from the vasprun.xml file that is returned by the bandstructure calculation, such as the bandstructure values and the k-points-path for which it was calculated. Additionally, it contains the Fermi energy, the index of the valence band and the

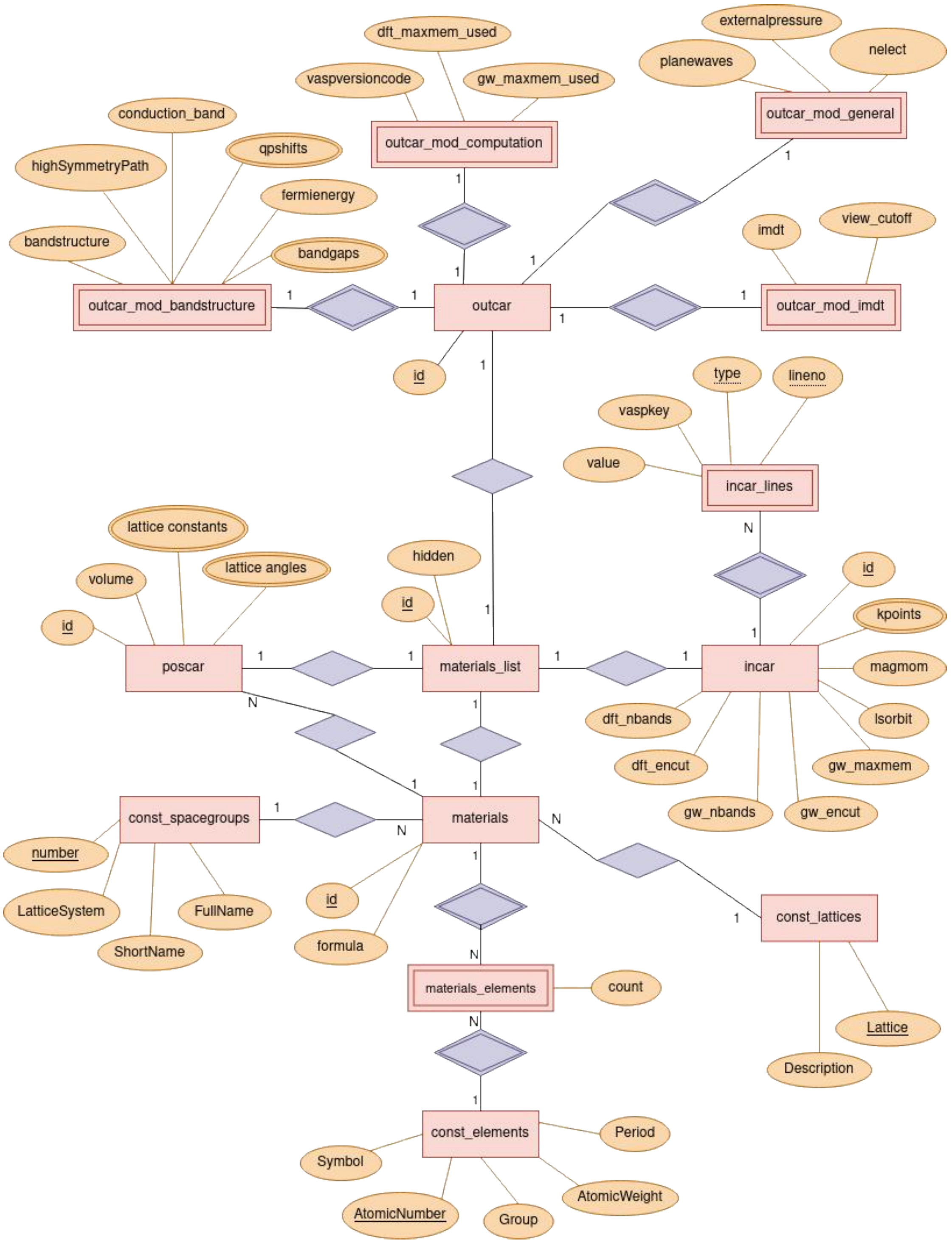


Figure 4.3: ER-diagram in Chen's notation[44] of the most important entities and attributes of the database schema. In Chen's notation, the '1's and 'N's signify if two entities are related in a 1:1, 1:N, or N:N relation. This image was created using Draw.io[45].

conduction band, the DFT-bandgap at the Γ point, as well the the smallest direct and smallest indirect DFT-bandgaps. It also contains data parsed from the corresponding G_0W_0 OUTCAR file, such as the three G_0W_0 bandgaps (smallest direct, smallest indirect, and at Γ), and the quasiparticle shifts for the valence band and the conduction band at different k-points. The table `outcar_mod_computation` contains information on how the calculation was executed, such as the VASP version and the number of cores used. The table `outcar_mod_general` contains the number of electrons (NELECT), the number of plane waves, and the external pressure. The table `outcar_mod_imdt` contains the information that is necessary to plot the inverse macroscopic dielectric tensor, taken from the G_0W_0 OUTCAR file.

In the table `materials`, each row represents one material, determined by chemical formula and spacegroup. Whenever new data is uploaded, a new row is added to `materials` if and only if the same combination of formula and spacegroup does not yet exist in the table. Additionally to formula and spacegroup, the table contains the abbreviation for the Bravais lattice variation[40] (e.g. RHL1, MCLC5) and each row is given a unique ID.

Finally, the table `materials_list` relates all these tables to each other: Each entry in this table contains a `material_id`, `poscar_id`, `incar_id`, and `outcar_id`, as well as one new unique ID for the whole row which is used to refer to the entire calculation. It also contains a boolean column called `hidden`, which can be set to true in order to prevent the entry from being shown to users.

These are the most central tables of the database. There are further tables which are less important and will not be described in detail here. A few of them are constant tables that are never changed, holding information such as all elements of the periodic table and their properties, or all 230 spacegroups and their shortcuts, full names, and numbers. Another table called `materials_elements` lists the the number of each element contained in the formula of a material, linked to the `materials` table with a `material_id`. This information is used to create the image of the periodic table on the frontend. The tables `references_bib` and `references_data` contain the references and citations that are shown on the frontend for some materials. The table `uploaded_files` contains the information about the files that were uploaded by users, including file name, user ID, and upload date.

4.3.2 Backend

The server-side part of the application, which includes the business logic as well as the connection between the user and the database, is written in PHP. This part of the program parses the files that are uploaded, and saves and retrieves the data in the database described in Section 4.3.1. It also implements the API that provides the data in JavaScript Object Notation[46] (JSON) form to the frontend or to external API queries.

The main part of the business logic is found in the package `classes`. In this package, there is a sub-package called `DB`, which contains all the code that is used to interface with the database. In `DB`, there is a class for every table that exists in the database, except in some cases where one class references two tables, such as `INCARDatabase` which interfaces with both the `incar` table and the `incar_lines` table, or `Lattice` which interfaces with both `const_lattices` and `poscar`. Each of these database classes has a counterpart in the `classes` package, which represents the object from the database table as a PHP class, having the same field variables as the table has columns. Each class in the `DB` package also contains a method to fetch an entry from the database table and return it as a PHP class. The database classes referencing non-constant tables also have a method to save the data from from the respective class in the database.

All the database classes which reference specific tables in the database extend the `Database` class. This class is responsible for the database access. It creates the database connection using the MySQLi[47] library. `Database` executes SQL-queries with the help of the classes `DatabaseBinding` and `DatabaseBindingSet`, which help with the handling of data types and inserting the actual values into the SQL-queries. The `GWDatabase` class extends the `Database` class and contains a number of methods that can be used to easily create SQL queries for any chosen table. These methods are used in the table-specific database classes, such as `INCARDatabase` or `MaterialDatabase`. These classes also contain methods to make more specific SQL queries.

Outside of the `DB` package are the counterparts to these table-specific database classes, such as `INCAR` or `Material`. Each of these classes holds the data from the table it corresponds to, e.g. the `Materials` class has the same field variables as the `materials` table has columns. The database classes transform the data they fetch from the MariaDB tables into these PHP classes, e.g. the method in `INCARDatabase` which fetches an entry from the `incar` table returns an instance of the `INCAR` class. These classes also implement classes for parsing the data from the uploaded INCAR, KPOINTS, POSCAR, OUTCAR, and vasprun.xml files. Of particular interest are the `GWEntryDatabase` class and the `GWEntry` class, which correspond to the `materials_list` table and thus reference the calculation as a whole. By having access to the ID of every sub-element of a calculation, these classes can connect all the data from the different tables together. It should be noted that the classes corresponding to the tables `outcar_mod_bandstructure`, `outcar_mod_computation`, `outcar_mod_general`, and `outcar_mod_imdt` are located in the package `modules` instead of `classes`.

There are other classes in the package `classes` which help with parsing and processing data, and structuring and navigating the website. The class `PeriodicTable` builds the periodic table image showing which elements are represented in the database. In the package `Structure`, there are classes that help with the parsing of POSCAR data, and in the `Searchbar` package are the classes that handle the search feature which will be described in Section 4.3.3.

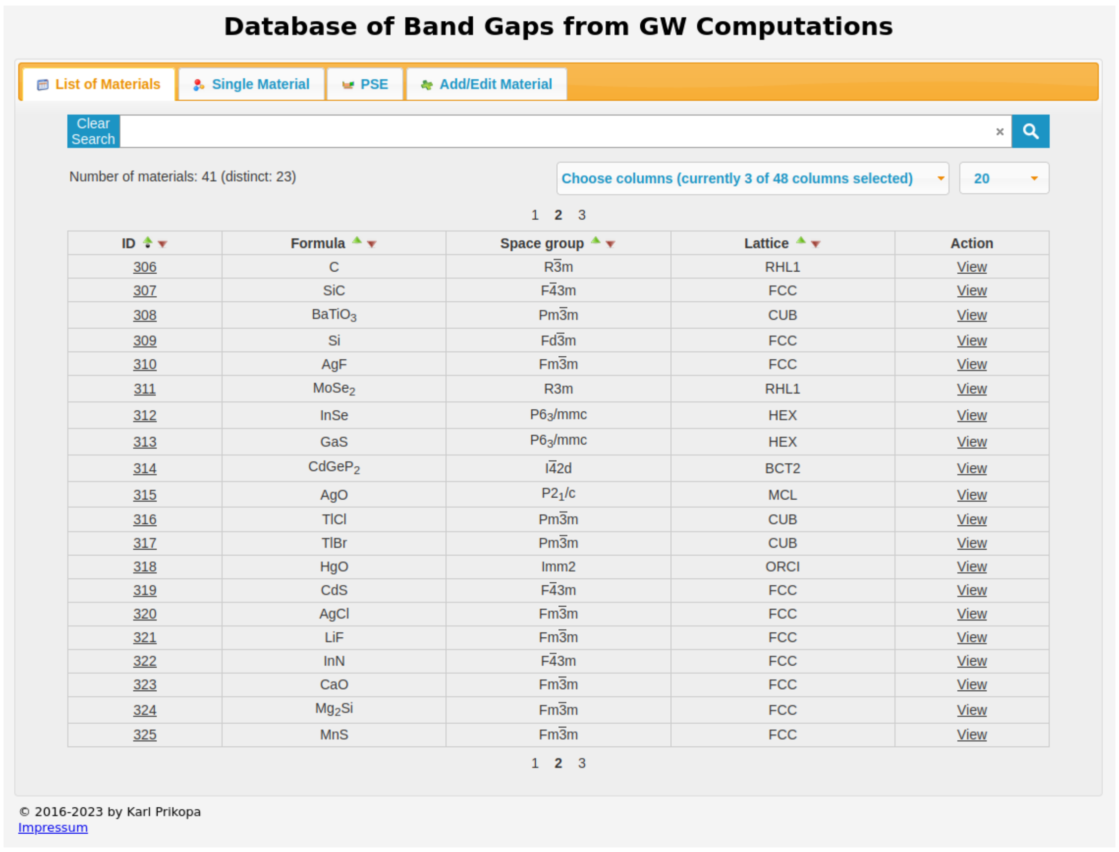
In the package `apps`, the file api.php contains the API that connects the backend and the frontend. A JavaScript method in the frontend can request data from the API, which is transferred as a JSON, or it can request that data be added or edited in the database. The API accepts both GET and POST requests, making no difference between them.

4.3.3 Frontend

To make the data universally and easily accessible, a web-application was developed. Users can use the graphical user interface (GUI) of the web-application to browse and search for data (Figure 4.4), look at the data of each calculation and download input- and output files (Figure 4.13 to Figure 4.17), and get an overview over which elements are in the database (Figure 4.18). Additionally, new data can be uploaded (Figure 4.19 to Figure 4.25). This part of the application is password-protected so that only high-quality data is uploaded. The website consists of four tabs: ‘List of Materials’, ‘Single Material’, ‘PSE’, and ‘Add/Edit Material’.

The ‘List of Materials’ tab, shown in Figure 4.4, shows a list of all calculations that were uploaded and are not hidden. Each entry of the list represents one calculation, or one element of the `materials_list` table of the MariaDB database. By default there are 20 entries on one page of the list, but the user can change this via a dropdown menu. In the ‘Choose Columns’ dropdown menu, shown in Figure 4.5, the user can choose which information should be shown in the list, such as bandgaps, lattice constants, or VASP version. In this way, data can be easily and quickly compared between calculations without having to open each entry separately. The user can choose any of the columns shown in the list as a sort column by clicking on the small arrows next to the column title.

The search bar can be used to search for data. By default, it searches for chemical formulae, as shown in Figure 4.6. For more complex search requests, a custom search syntax has been implemented. It may not be immediately obvious how it works at


Database of Band Gaps from GW Computations
List of Materials
Single Material
PSE
Add/Edit Material
Clear Search
Choose columns (currently 3 of 48 columns selected)
20
1 2 3

Number of materials: 41 (distinct: 23)

ID	Formula	Space group	Lattice	Action
306	C	$R\bar{3}m$	RHL1	View
307	SiC	$F\bar{4}3m$	FCC	View
308	$BaTiO_3$	$Pm\bar{3}m$	CUB	View
309	Si	$Fd\bar{3}m$	FCC	View
310	AgF	$Fm\bar{3}m$	FCC	View
311	$MoSe_2$	R3m	RHL1	View
312	InSe	$P6_3/mmc$	HEX	View
313	GaS	$P6_3/mmc$	HEX	View
314	$CdGeP_2$	$I\bar{4}2d$	BCT2	View
315	AgO	$P2_1/c$	MCL	View
316	TlCl	$Pm\bar{3}m$	CUB	View
317	TlBr	$Pm\bar{3}m$	CUB	View
318	HgO	Imm2	ORCI	View
319	CdS	$F\bar{4}3m$	FCC	View
320	AgCl	$Fm\bar{3}m$	FCC	View
321	LiF	$Fm\bar{3}m$	FCC	View
322	InN	$F\bar{4}3m$	FCC	View
323	CaO	$Fm\bar{3}m$	FCC	View
324	Mg_2Si	$Fm\bar{3}m$	FCC	View
325	MnS	$Fm\bar{3}m$	FCC	View

© 2016-2023 by Karl Prikopa
Impressum

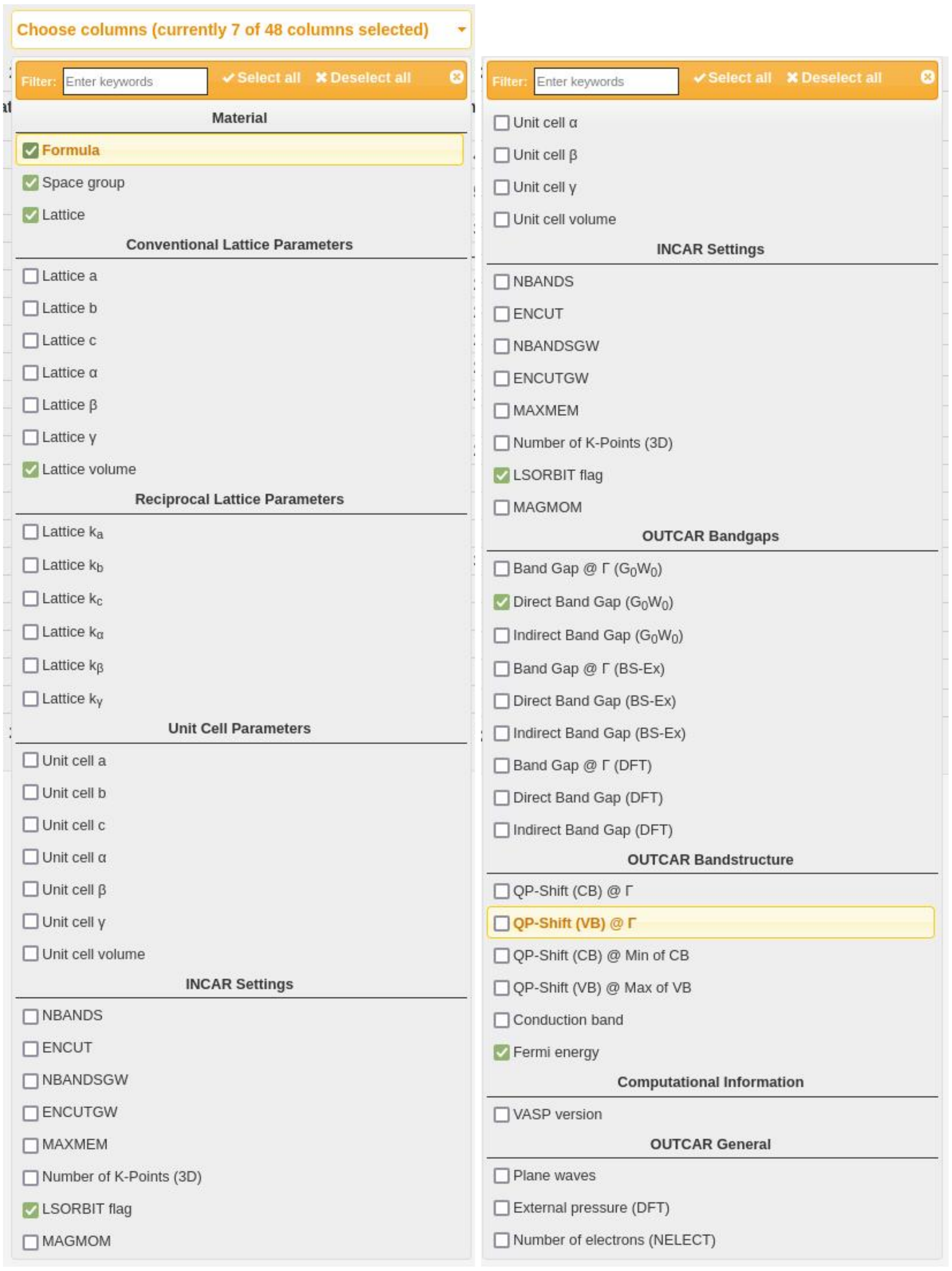
Figure 4.4: The 'List of Materials' tab on the GW-Database website.

first glance, so for documentation purposes, some examples will be provided in the following paragraphs.

If a searchable keyword is followed by a colon, then this category can be searched specifically. The available search-keywords are shown as suggestions when typing into the search bar, as can be seen in Figure 4.7. It should be noted that new search-keywords can easily be added in the `GWDBApp` class in the `/classes/Searchbar` package, simply by providing the keyword and the name of the column in the database that should be searched, and so the suggestions shown in Figure 4.7 are subject to change as the application continues to be adapted to the preferences of the users.

Note that the following screenshots are only of the search functionality, and are not indicative of the materials that are available in the GW-Database, because they were taken using a locally running instance of the database, which contained only a small subset of the materials at the time. This is due to having to reset the database multiple times during development. For this reason, the searches shown in the screenshots produce fewer results than would be available in the actual GW-Database.

Figure 4.8 shows the difference in results when searching for 'F' while specifying the spacegroup, in contrast to Figure 4.6 where spacegroup was not specified. Fig-


Choose columns (currently 7 of 48 columns selected)

Material

- ✓ Formula
- ✓ Space group
- ✓ Lattice

Conventional Lattice Parameters

- ☐ Lattice a
- ☐ Lattice b
- ☐ Lattice c
- ☐ Lattice α
- ☐ Lattice β
- ☐ Lattice γ
- ✓ Lattice volume

Reciprocal Lattice Parameters

- ☐ Lattice k_a
- ☐ Lattice k_b
- ☐ Lattice k_c
- ☐ Lattice k_α
- ☐ Lattice k_β
- ☐ Lattice k_γ

Unit Cell Parameters

- ☐ Unit cell a
- ☐ Unit cell b
- ☐ Unit cell c
- ☐ Unit cell α
- ☐ Unit cell β
- ☐ Unit cell γ
- ☐ Unit cell volume

INCAR Settings

- ☐ NBANDS
- ☐ ENCUT
- ☐ NBANDSGW
- ☐ ENCUTGW
- ☐ MAXMEM
- ☐ Number of K-Points (3D)
- ✓ LSORBIT flag
- ☐ MAGMOM

- ☐ Unit cell α
- ☐ Unit cell β
- ☐ Unit cell γ
- ☐ Unit cell volume

INCAR Settings

- ☐ NBANDS
- ☐ ENCUT
- ☐ NBANDSGW
- ☐ ENCUTGW
- ☐ MAXMEM
- ☐ Number of K-Points (3D)
- ✓ LSORBIT flag
- ☐ MAGMOM

OUTCAR Bandgaps

- ☐ Band Gap @ Γ (G_0W_0)
- ✓ Direct Band Gap (G_0W_0)
- ☐ Indirect Band Gap (G_0W_0)
- ☐ Band Gap @ Γ (BS-Ex)
- ☐ Direct Band Gap (BS-Ex)
- ☐ Indirect Band Gap (BS-Ex)
- ☐ Band Gap @ Γ (DFT)
- ☐ Direct Band Gap (DFT)
- ☐ Indirect Band Gap (DFT)

OUTCAR Bandstructure

- ☐ QP-Shift (CB) @ Γ
- ☐ QP-Shift (VB) @ Γ
- ☐ QP-Shift (CB) @ Min of CB
- ☐ QP-Shift (VB) @ Max of VB
- ☐ Conduction band
- ✓ Fermi energy

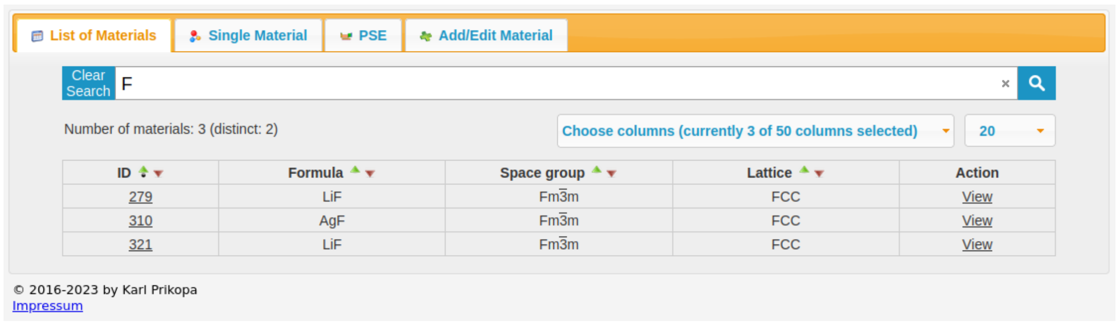
Computational Information

- ☐ VASP version

OUTCAR General

- ☐ Plane waves
- ☐ External pressure (DFT)
- ☐ Number of electrons (NELECT)

Figure 4.5: The dropdown menu where the user can choose which columns should be shown in the list of all calculations. The screenshot has been re-formatted into two columns in order for the image to fit the page.



F

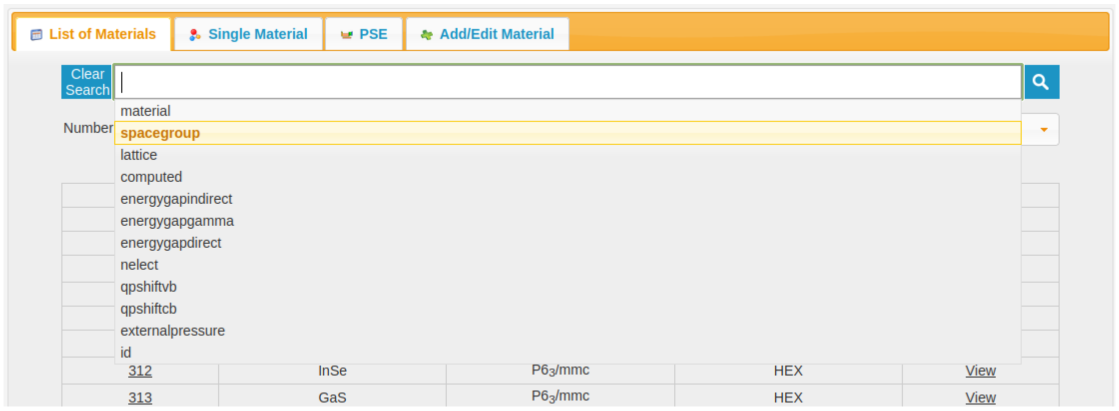
Number of materials: 3 (distinct: 2)

Choose columns (currently 3 of 50 columns selected) 20

ID	Formula	Space group	Lattice	Action
279	LiF	Fm$\bar{3}$m	FCC	View
310	AgF	Fm$\bar{3}$m	FCC	View
321	LiF	Fm$\bar{3}$m	FCC	View

© 2016-2023 by Karl Prikopa

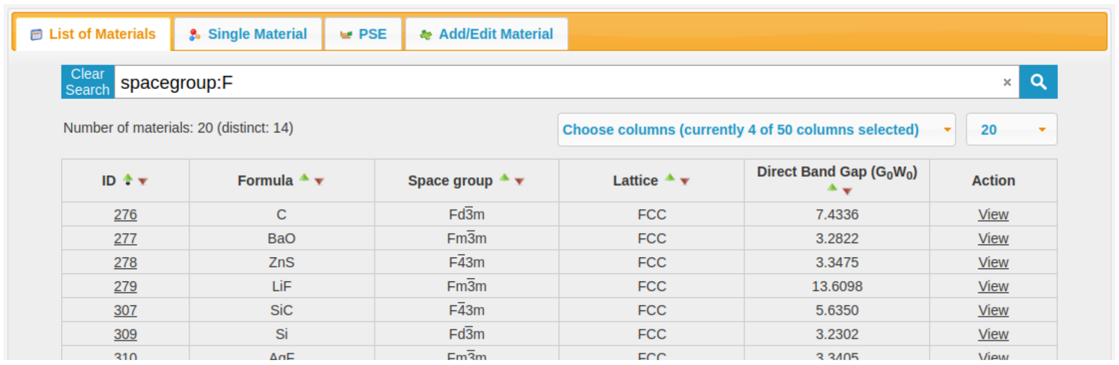
Figure 4.6: Searching for 'F' in the search bar gives as results all entries where the chemical formula contains fluorine.



312	InSe	P6_3/mmc	HEX	View
313	GaS	P6_3/mmc	HEX	View

Figure 4.7: Suggestions for available search-keywords.

ure 4.9 shows how to search for a specific spacegroup. Subscript numbers should be represented by preceding the number with an underscore ('_'), and numbers with a bar above them should be represented by preceding the number with a minus or hyphen symbol ('-'). It should be noted that there cannot be a space between the colon and the search value, or this search will not work.



spacegroup:F

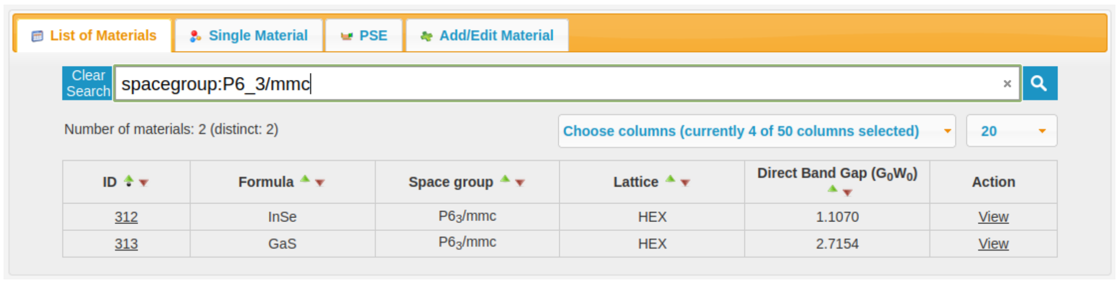
Number of materials: 20 (distinct: 14)

Choose columns (currently 4 of 50 columns selected) 20

ID	Formula	Space group	Lattice	Direct Band Gap (G_0W_0)	Action
276	C	Fd$\bar{3}$m	FCC	7.4336	View
277	BaO	Fm$\bar{3}$m	FCC	3.2822	View
278	ZnS	F$\bar{4}$3m	FCC	3.3475	View
279	LiF	Fm$\bar{3}$m	FCC	13.6098	View
307	SiC	F$\bar{4}$3m	FCC	5.6350	View
309	Si	Fd$\bar{3}$m	FCC	3.2302	View
310	AgF	Fm$\bar{3}$m	FCC	3.3405	View

Figure 4.8: A search for all calculations where the spacegroup contains the letter 'F'.

To search for numeric values like bandgaps, the available operations are 'equals' (=), 'less than'/'greater than' (< / >), 'less or equal'/'greater or equal' (<= / >=)



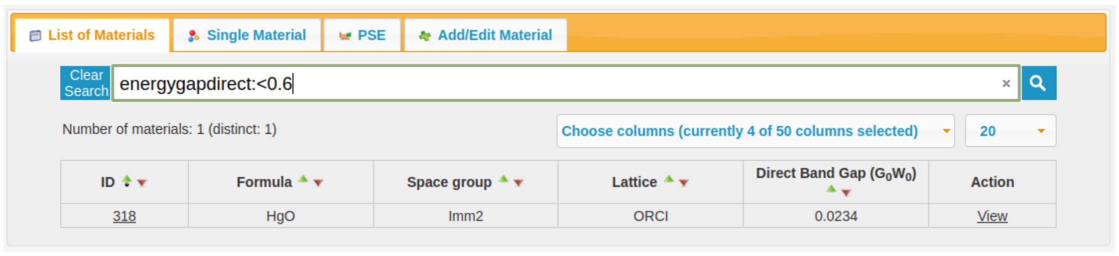
spacegroup:P6_3/mmc

Number of materials: 2 (distinct: 2)

ID	Formula	Space group	Lattice	Direct Band Gap (G_0W_0)	Action
312	InSe	$P6_3/mmc$	HEX	1.1070	View
313	GaS	$P6_3/mmc$	HEX	2.7154	View

Figure 4.9: A search for a calculations where the spacegroup is $P6_3/mmc$.

and 'between' (<>) are available. Figure 4.10 and Figure 4.11 show the syntax for conducting a 'less than' and a 'between' search, respectively. Again, it is important that there is no space between the colon and the operation, nor between the operation and the value.

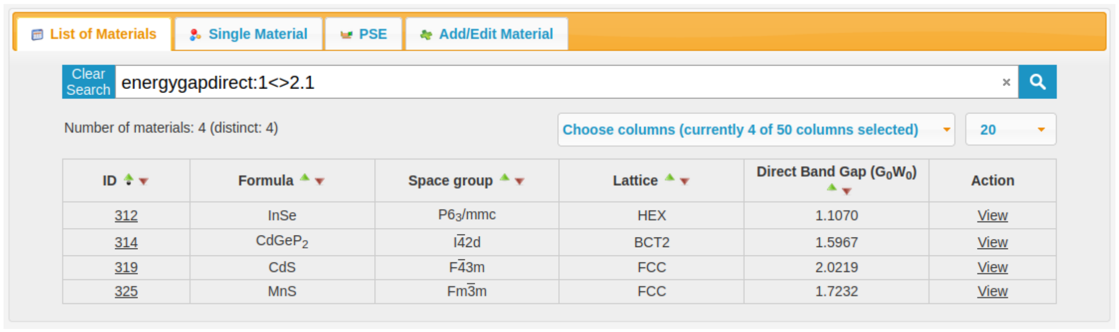


energygapdirect:<0.6

Number of materials: 1 (distinct: 1)

ID	Formula	Space group	Lattice	Direct Band Gap (G_0W_0)	Action
318	HgO	Imm2	ORCI	0.0234	View

Figure 4.10: A search for all calculations resulting in a direct bandgap smaller than 0.6 eV.



energygapdirect:1<>2.1

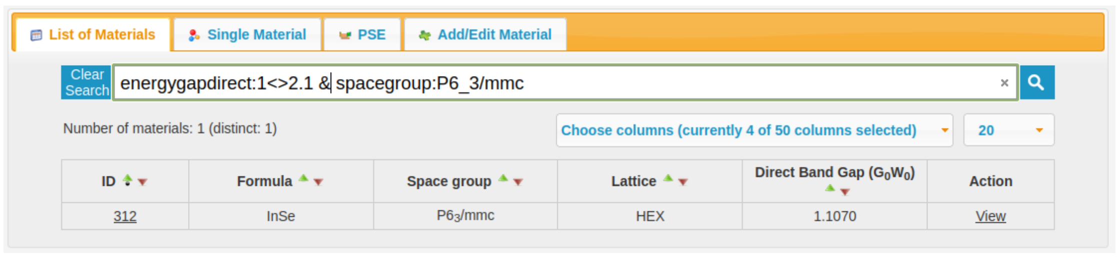
Number of materials: 4 (distinct: 4)

ID	Formula	Space group	Lattice	Direct Band Gap (G_0W_0)	Action
312	InSe	$P6_3/mmc$	HEX	1.1070	View
314	$CdGeP_2$	$I\bar{4}2d$	BCT2	1.5967	View
319	CdS	$F\bar{4}3m$	FCC	2.0219	View
325	MnS	$Fm\bar{3}m$	FCC	1.7232	View

Figure 4.11: A search for all calculations resulting in a direct bandgap between 1 eV and 2.1 eV.

Using an '&' symbol, a logical AND search can be executed, as seen in Figure 4.12. On the other hand, a logical OR search is currently not implemented. Instead of an '&' symbol, a space can also be used for the same result.

When clicking on an ID in the list, the 'Single Material' tab is opened. It contains all available information about the calculation associated with the ID. At the top of the page (shown in Figure 4.13) is the table of contents, followed by general information about the material, such the formula and lattice parameters, and an interactive visualization of the unit cell created using JSmol[48]. Below that are the



energygapdirect:1<>2.1 & spacegroup:P6_3/mmc

Number of materials: 1 (distinct: 1)

Choose columns (currently 4 of 50 columns selected) | 20

ID	Formula	Space group	Lattice	Direct Band Gap (G_0W_0)	Action
312	InSe	$P6_3/mmc$	HEX	1.1070	View

Figure 4.12: A search for all calculations which result in a direct bandgap between 1 eV and 2.1 eV and where the spacegroup is $P6_3/mmc$.

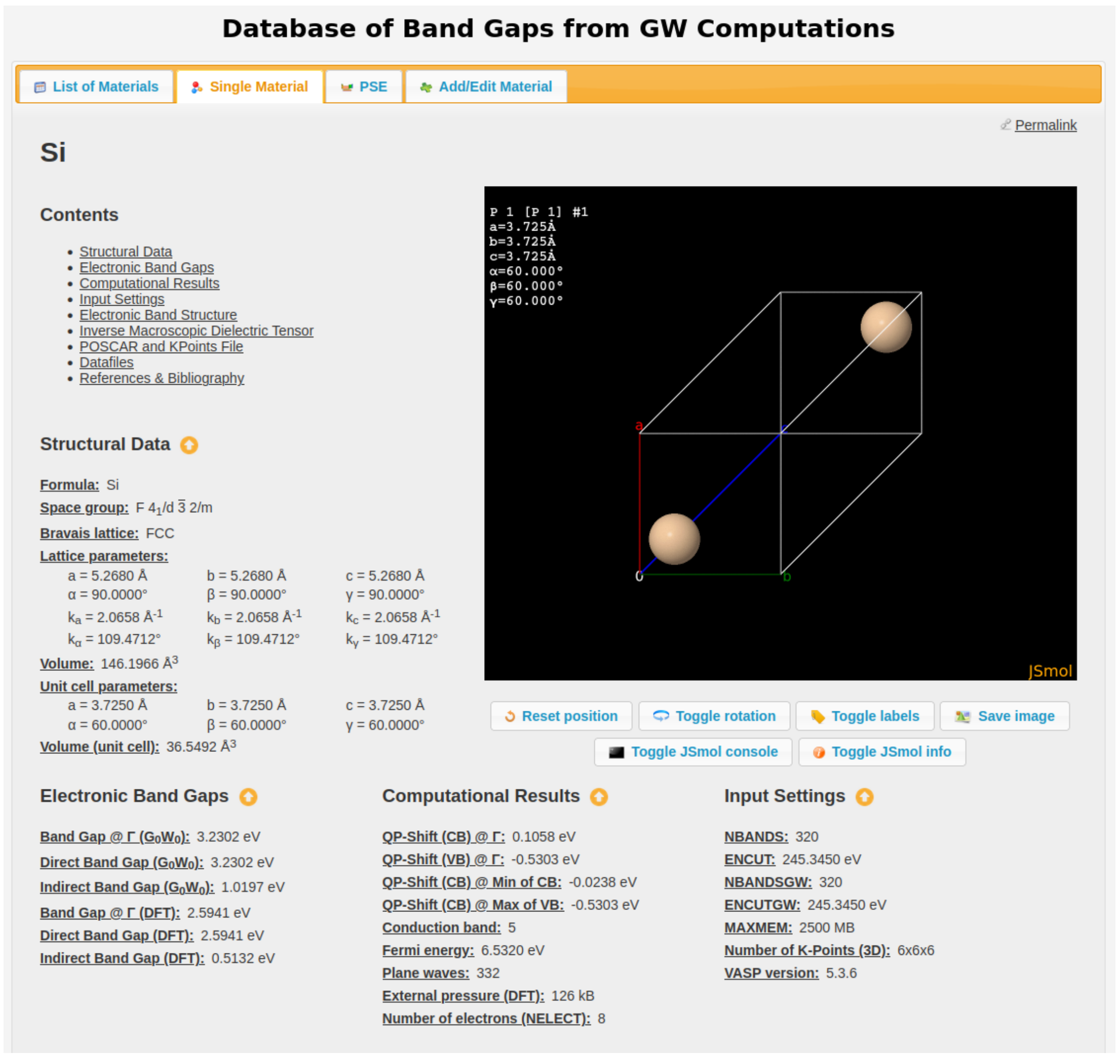
most important results and computational parameters. The smallest direct and smallest indirect bandgaps calculated by both DFT and GW are presented here, as well as the bandgaps at the Γ k-point for both methods.

If a bandstructure calculation has been executed and its resulting vasprun.xml file has been uploaded, a plot of the bandstructure is created and shown on the website (Figure 4.14). It shows the energy bands of the material along the high-symmetry k-points path that was used in the bandstructure calculation. The plot is constructed from a DFT bandstructure calculation, rather than a GW bandstructure calculation, because performing a bandstructure calculation with GW would be difficult due to the very high computational cost.

Scrolling down further reveals a plot of the relative permittivity and the optical conductivity (Figure 4.15), the data for which is parsed from the information about the inverse macroscopic dielectric tensor (IMDT) that is written in the GW OUTCAR. Permittivity is dependent on the direction in the material, which is why this information is output in the form of a tensor by VASP. In order to plot it on the website, the average value over all directions is used. The optical conductivity is then calculated from the imaginary part of the permittivity.

To make it possible to reproduce the calculations, the input and output files are provided for download, as shown in Figure 4.16. This includes the POSCAR file, the KPOINTS file for the bandstructure calculation, and the OUTCAR and vasprun.xml files for step 1 to step 4 described in Section 4.2. Also visible in this figure is a visualization of the Brillouin zone showing the high-symmetry k-points and a suggested path between them[40]. For some materials, reference data from other publications is available. In these cases, the data and citations can be found on the bottom of the Single Material tab (Figure 4.17).

Figure 4.18 shows the 'PSE' tab. It is a periodic table that visualizes how many different materials in the database contain each element. When the 'Update' button is clicked, the image is re-calculated, taking into account any new materials that may


Database of Band Gaps from GW Computations

Si

Contents

- Structural Data
- Electronic Band Gaps
- Computational Results
- Input Settings
- Electronic Band Structure
- Inverse Macroscopic Dielectric Tensor
- POSCAR and KPoints File
- Datafiles
- References & Bibliography

Structural Data

Formula: Si
Space group: F 4_1/d $\bar{3}$ 2/m
Bravais lattice: FCC
Lattice parameters:

a = 5.2680 Å	b = 5.2680 Å	c = 5.2680 Å
α = 90.0000°	β = 90.0000°	γ = 90.0000°
k_a = 2.0658 $Å^{-1}$	k_b = 2.0658 $Å^{-1}$	k_c = 2.0658 $Å^{-1}$
k_α = 109.4712°	k_β = 109.4712°	k_γ = 109.4712°

Volume: 146.1966 $Å^3$
Unit cell parameters:

a = 3.7250 Å	b = 3.7250 Å	c = 3.7250 Å
α = 60.0000°	β = 60.0000°	γ = 60.0000°

Volume (unit cell): 36.5492 $Å^3$

```
P 1  [P 1] #1
a=3.725Å
b=3.725Å
c=3.725Å
α=60.000°
β=60.000°
γ=60.000°
```

Electronic Band Gaps

Band Gap @ Γ (G_0W_0): 3.2302 eV
Direct Band Gap (G_0W_0): 3.2302 eV
Indirect Band Gap (G_0W_0): 1.0197 eV
Band Gap @ Γ (DFT): 2.5941 eV
Direct Band Gap (DFT): 2.5941 eV
Indirect Band Gap (DFT): 0.5132 eV

Computational Results

QP-Shift (CB) @ Γ: 0.1058 eV
QP-Shift (VB) @ Γ: -0.5303 eV
QP-Shift (CB) @ Min of CB: -0.0238 eV
QP-Shift (CB) @ Max of VB: -0.5303 eV
Conduction band: 5
Fermi energy: 6.5320 eV
Plane waves: 332
External pressure (DFT): 126 kB
Number of electrons (NELECT): 8

Input Settings

NBANDS: 320
ENCUT: 245.3450 eV
NBANDSGW: 320
ENCUTGW: 245.3450 eV
MAXMEM: 2500 MB
Number of K-Points (3D): 6x6x6
VASP version: 5.3.6

Figure 4.13: The top of the 'Single Material' tab.

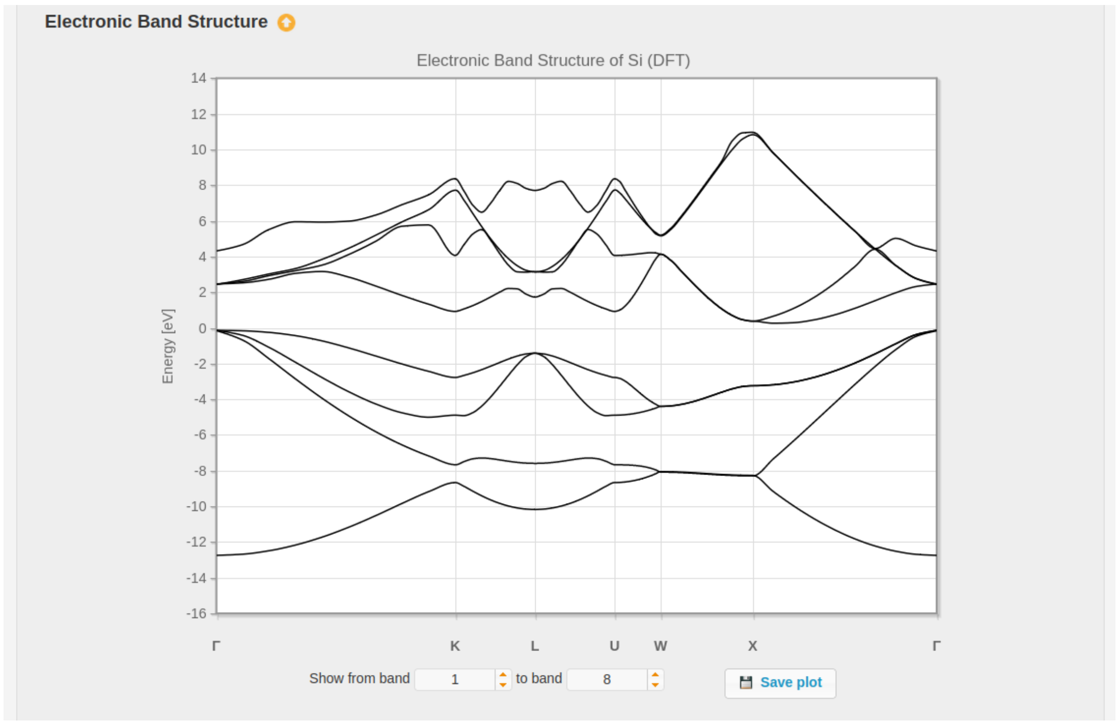

Electronic Band Structure
Electronic Band Structure of Si (DFT)
Energy [eV]
Γ
K
L
U
W
X
Γ
Show from band
1
to band
8
Save plot

Figure 4.14: The plot of the bandstructure in the 'Single Material' tab.

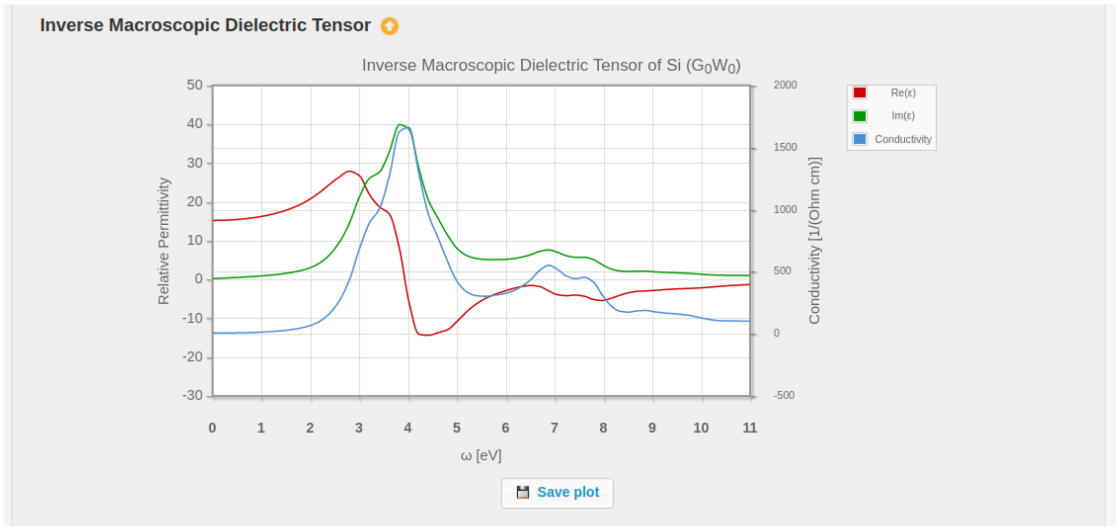
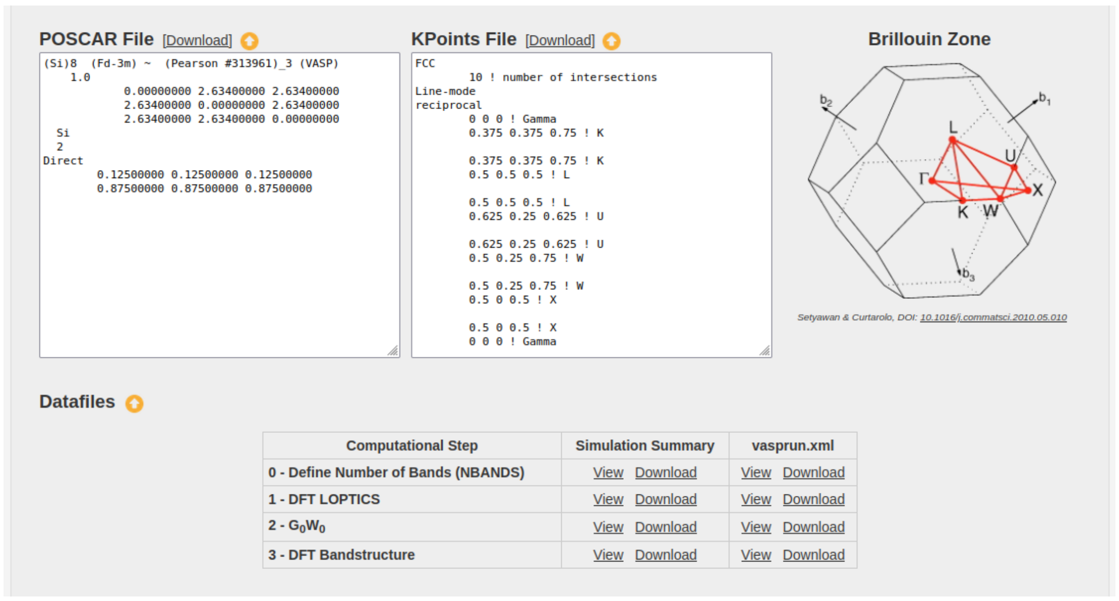

Inverse Macroscopic Dielectric Tensor
Inverse Macroscopic Dielectric Tensor of Si (G_0W_0)
Relative Permittivity
Conductivity [1/(Ohm cm)]
ω [eV]
Re(ε)
Im(ε)
Conductivity
Save plot

Figure 4.15: The plot of the permittivity and optical conductivity in the 'Single Material' tab.



POSCAR File [Download]

```
(Si)8  (Fd-3m) ~  (Pearson #313961)_3 (VASP)
    1.0
            0.00000000 2.63400000 2.63400000
            2.63400000 0.00000000 2.63400000
            2.63400000 2.63400000 0.00000000
  Si
  2
Direct
        0.12500000 0.12500000 0.12500000
        0.87500000 0.87500000 0.87500000
```

KPoints File [Download]

```
FCC
        10 ! number of intersections
Line-mode
reciprocal
        0 0 0 ! Gamma
        0.375 0.375 0.75 ! K

        0.375 0.375 0.75 ! K
        0.5 0.5 0.5 ! L

        0.5 0.5 0.5 ! L
        0.625 0.25 0.625 ! U

        0.625 0.25 0.625 ! U
        0.5 0.25 0.75 ! W

        0.5 0.25 0.75 ! W
        0.5 0 0.5 ! X

        0.5 0 0.5 ! X
        0 0 0 ! Gamma
```

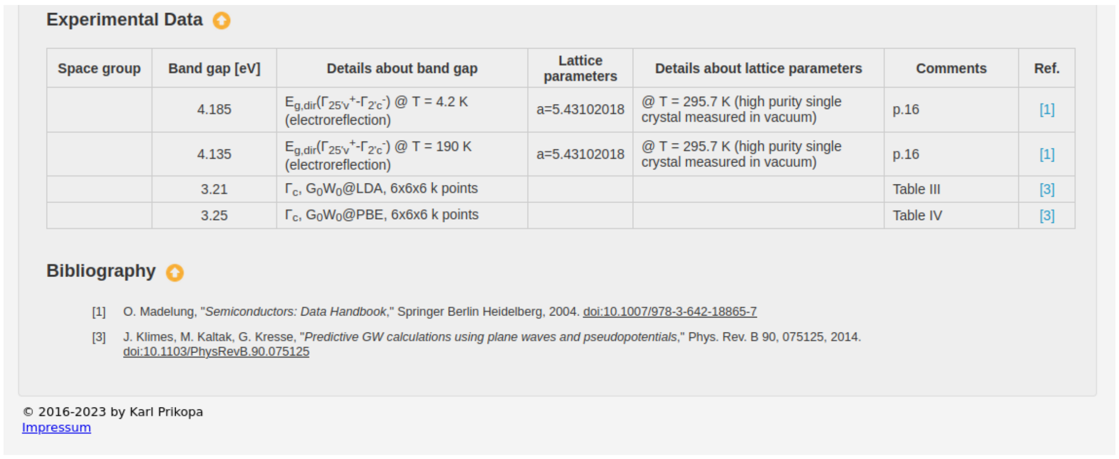
Brillouin Zone

Setyawan & Curtarolo, DOI: 10.1016/j.commatsci.2010.05.010

Datafiles

Computational Step	Simulation Summary	vasprun.xml
0 - Define Number of Bands (NBANDS)	View Download	View Download
1 - DFT LOPTICS	View Download	View Download
2 - G_0W_0	View Download	View Download
3 - DFT Bandstructure	View Download	View Download

Figure 4.16: The input and output files that can be downloaded in the 'Single Material' tab.



Experimental Data

Space group	Band gap [eV]	Details about band gap	Lattice parameters	Details about lattice parameters	Comments	Ref.
	4.185	$E_{g,dir}(\Gamma_{25'v}{}^{+}-\Gamma_{2'c}{}^{-})$ @ T = 4.2 K (electroreflection)	a=5.43102018	@ T = 295.7 K (high purity single crystal measured in vacuum)	p.16	[1]
	4.135	$E_{g,dir}(\Gamma_{25'v}{}^{+}-\Gamma_{2'c}{}^{-})$ @ T = 190 K (electroreflection)	a=5.43102018	@ T = 295.7 K (high purity single crystal measured in vacuum)	p.16	[1]
	3.21	Γ_c, G_0W_0@LDA, 6x6x6 k points			Table III	[3]
	3.25	Γ_c, G_0W_0@PBE, 6x6x6 k points			Table IV	[3]

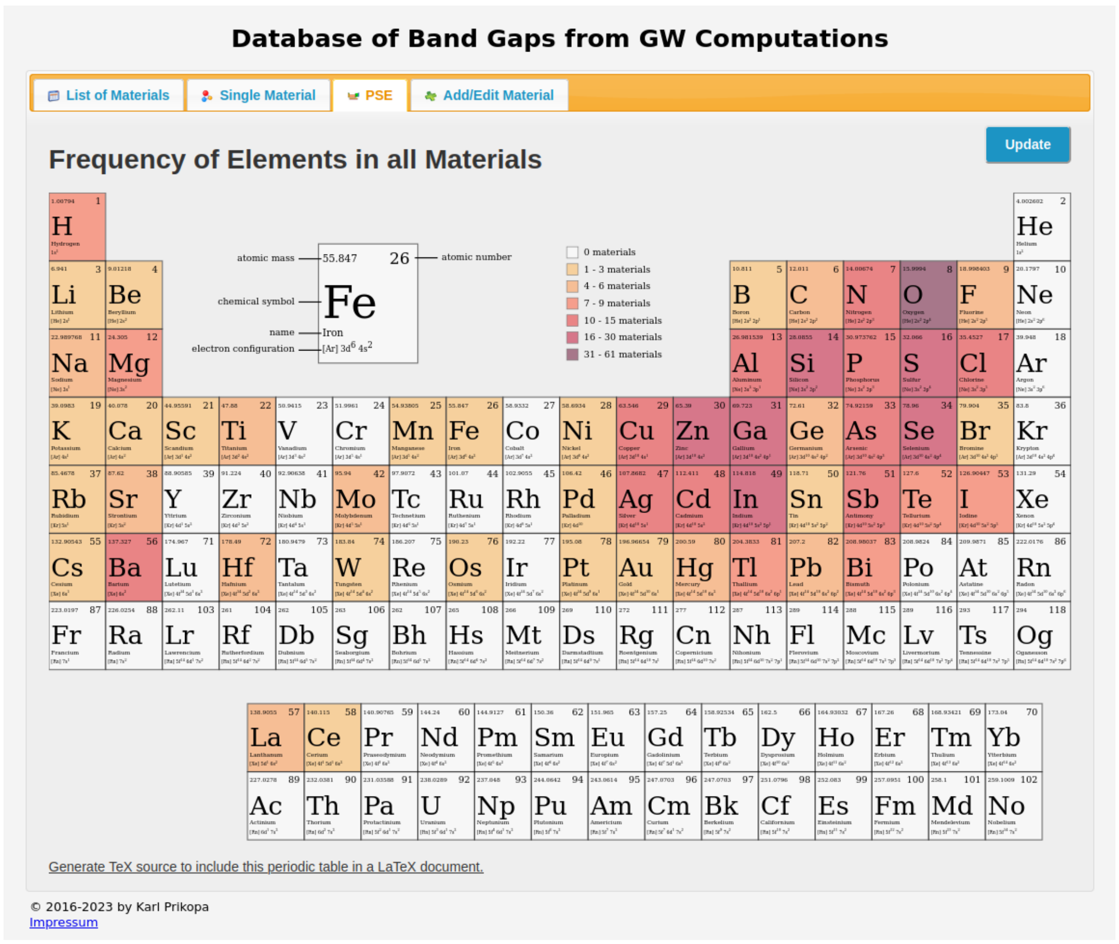
Bibliography

[1] O. Madelung, "*Semiconductors: Data Handbook*," Springer Berlin Heidelberg, 2004. doi:10.1007/978-3-642-18865-7

[3] J. Klimes, M. Kaltak, G. Kresse, "*Predictive GW calculations using plane waves and pseudopotentials*," Phys. Rev. B 90, 075125, 2014. doi:10.1103/PhysRevB.90.075125

Figure 4.17: Reference data and citations for the material in the 'Single Material' tab.

have been added since the last time it was updated. Adding the same materials multiple times does not change the frequency of the elements.


Database of Band Gaps from GW Computations

List of Materials
Single Material
PSE
Add/Edit Material
Update

Frequency of Elements in all Materials

atomic mass
55.847
26
atomic number
chemical symbol
Fe
name
Iron
electron configuration
[Ar] $3d^6$ $4s^2$

☐	0 materials
▨	1 - 3 materials
▨	4 - 6 materials
▨	7 - 9 materials
▨	10 - 15 materials
▨	16 - 30 materials
▨	31 - 61 materials

H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar K Ca Sc Ti V Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I Xe Cs Ba Lu Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn Fr Ra Lr Rf Db Sg Bh Hs Mt Ds Rg Cn Nh Fl Mc Lv Ts Og
La Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb
Ac Th Pa U Np Pu Am Cm Bk Cf Es Fm Md No

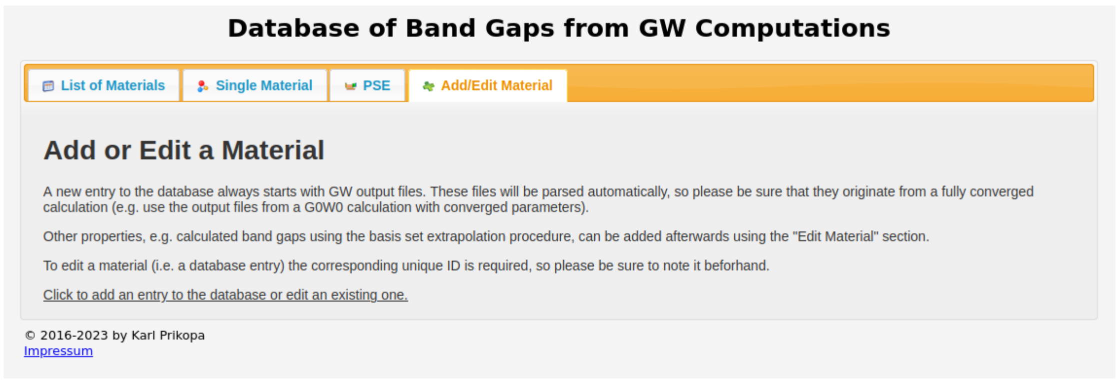
Generate TeX source to include this periodic table in a LaTeX document.

© 2016-2023 by Karl Prikopa
Impressum

Figure 4.18: The ‘PSE’ tab of the GW-Database website, showing current element frequencies.

The last tab is called ‘Add/Edit Material’ (Figure 4.19). As the name implies, here new materials can be added, and existing materials can be edited or hidden. It is desirable that only authorized users should be able to upload or edit data, which is why this tab is password-protected, as shown in Figure 4.20. This is only a rudimentary password protection, as a person with more expertise in cybersecurity would be needed to ensure it is truly secure. Fortunately, a low threat model is predicted for the GW-Database, and at this stage the password prompt should suffice to keep unauthorized persons from accessing the page.

Once the correct password has been entered, access to the page shown in Figure 4.21 is granted. To add data, at least one POSCAR, one KPOINTS, and one OUTCAR or vasprun.xml file is needed. However, to allow users to download all the files and see all available information, it is recommended to upload all files that can be parsed by the GW-Database. This means one POSCAR file, one KPOINTS file, and an OUTCAR and vasprun.xml file for each of the steps listed in Section 4.2. It is not



Add or Edit a Material

A new entry to the database always starts with GW output files. These files will be parsed automatically, so please be sure that they originate from a fully converged calculation (e.g. use the output files from a G0W0 calculation with converged parameters).

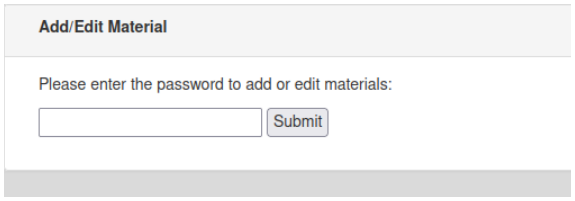
Other properties, e.g. calculated band gaps using the basis set extrapolation procedure, can be added afterwards using the "Edit Material" section.

To edit a material (i.e. a database entry) the corresponding unique ID is required, so please be sure to note it beforhand.

Click to add an entry to the database or edit an existing one.

© 2016-2023 by Karl Prikopa
Impressum

Figure 4.19: The ‘Add/Edit Material’ tab.



Please enter the password to add or edit materials:

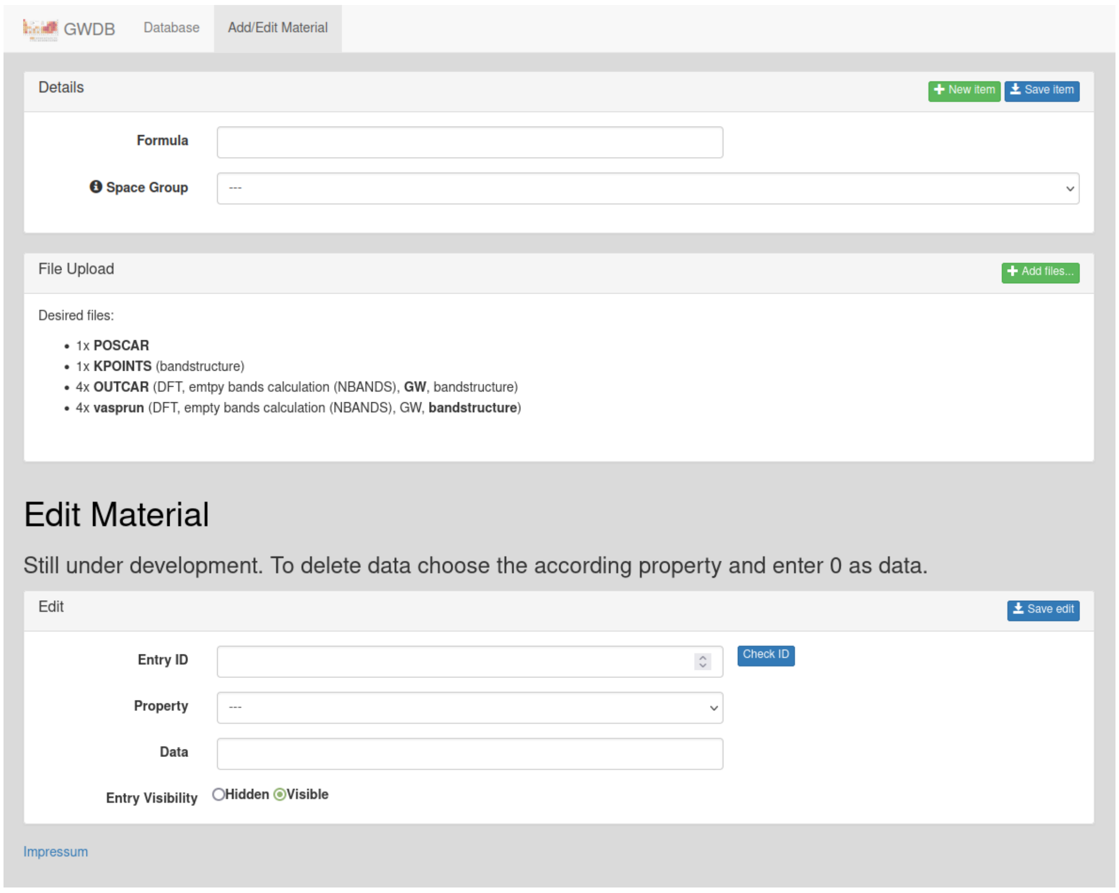
Figure 4.20: The password prompt for accessing the ‘Add/Edit Material’ tab.

necessary to upload the INCAR file, since all the information from the INCAR is available in the respective OUTCAR file.

The user also needs to select for each file what kind of file it is, type in the formula of the material, and choose the spacegroup from a dropdown menu (sorted by spacegroup number). Figure 4.22 shows the upload form when it is filled out. As soon as the ‘Save Item’ button is clicked, the data is parsed and added to the database, and can then be found in the ‘List of Materials’ tab.

After a calculation has been uploaded it can be edited in the ‘Edit Material’ section of the page. This feature was implemented in order to be able to include bandgaps that were calculated using the basis set extrapolation (BS-EX) method described in Section 2.2, where the calculation of the corrected bandgaps takes place outside of VASP calculations and cannot be parsed from the uploaded files. Additionally, in this section, entries can be set to be hidden or visible. If an entry is set to hidden, it cannot be seen in the ‘List of Materials’ tab or the ‘Single Material’ tab. If a calculation turns out to be inaccurate it can be removed from view in this way.

This ‘Edit Material’ section is a relatively new addition to the database and therefore the GUI is not very user-friendly yet. For instance, data is deleted by typing 0 into the data field, and there is no list of entries from which an ID can be chosen. To mitigate the latter issue, the ‘Check ID’ button exists. As shown in Figure 4.23, clicking this button allows the user to check whether they typed in the correct ID, by showing the formula and spacegroup of the material the ID refers to. Clicking on the link leads to the respective calculation on the ‘Single Material’ tab.



Desired files:

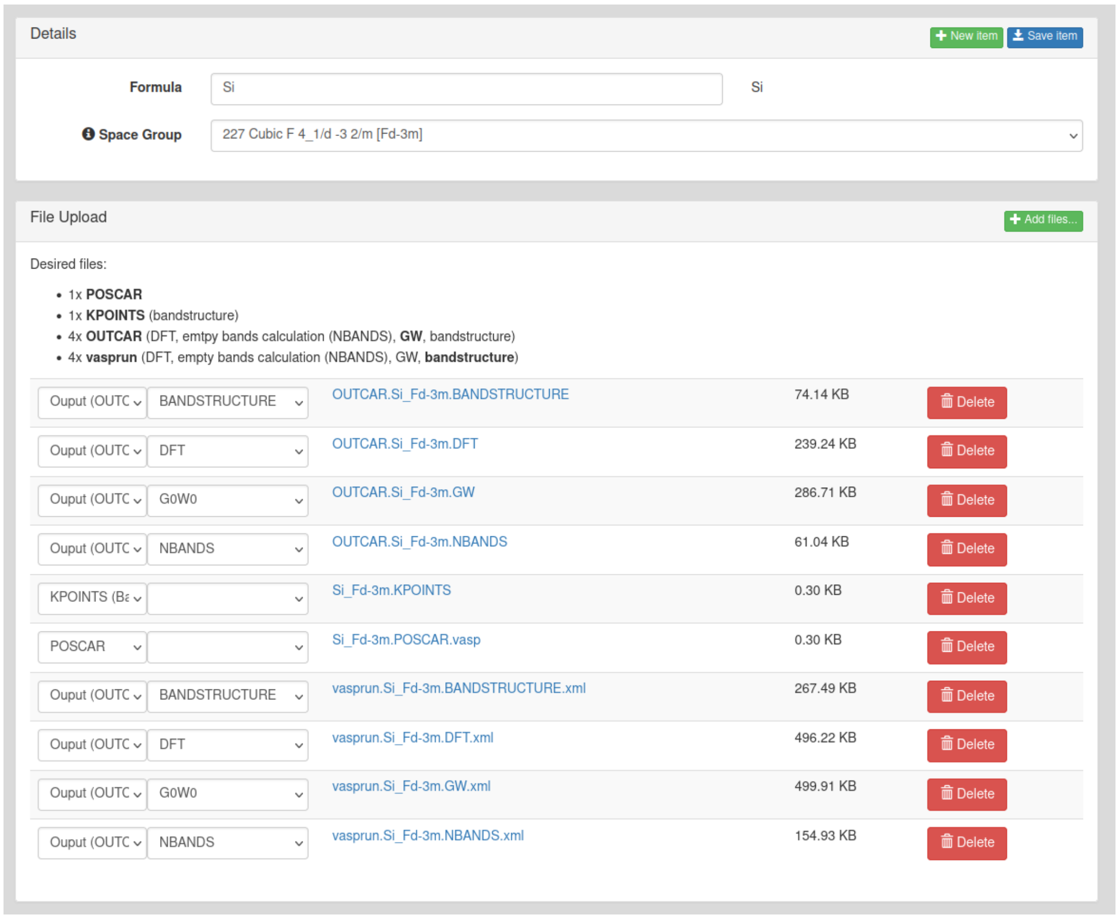
- 1x **POSCAR**
- 1x **KPOINTS** (bandstructure)
- 4x **OUTCAR** (DFT, emtpy bands calculation (NBANDS), **GW**, bandstructure)
- 4x **vasprun** (DFT, empty bands calculation (NBANDS), GW, **bandstructure**)

Edit Material

Still under development. To delete data choose the according property and enter 0 as data.

Entry Visibility Hidden [✓ Visible]

Figure 4.21: The ‘Add/Edit Material’ tab after having input the correct password.



Formula	Si	Si
Space Group	227 Cubic F 4_1/d -3 2/m [Fd-3m]	

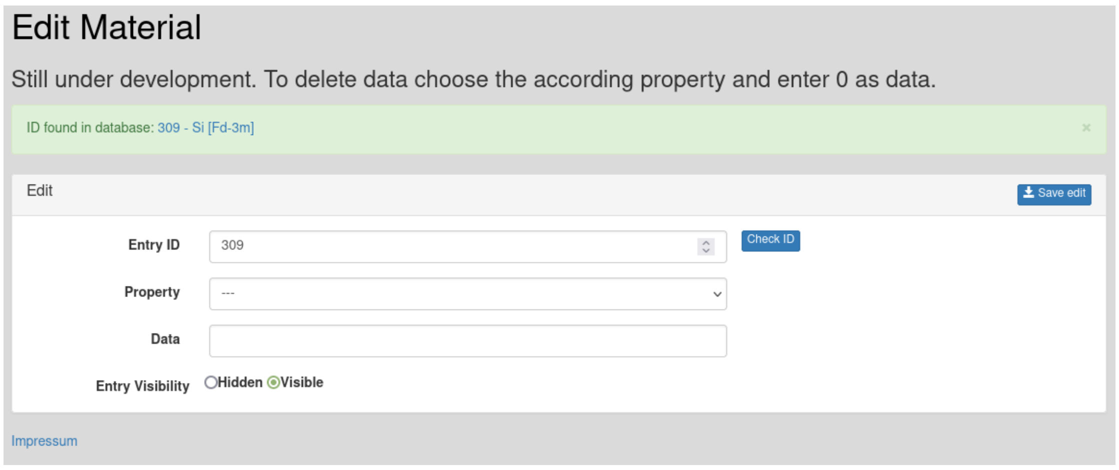
Desired files:

- 1x **POSCAR**
- 1x **KPOINTS** (bandstructure)
- 4x **OUTCAR** (DFT, emtpy bands calculation (NBANDS), **GW**, bandstructure)
- 4x **vasprun** (DFT, empty bands calculation (NBANDS), GW, **bandstructure**)

Ouput (OUTC	BANDSTRUCTURE	OUTCAR.Si_Fd-3m.BANDSTRUCTURE	74.14 KB	Delete
Ouput (OUTC	DFT	OUTCAR.Si_Fd-3m.DFT	239.24 KB	Delete
Ouput (OUTC	G0W0	OUTCAR.Si_Fd-3m.GW	286.71 KB	Delete
Ouput (OUTC	NBANDS	OUTCAR.Si_Fd-3m.NBANDS	61.04 KB	Delete
KPOINTS (Bε		Si_Fd-3m.KPOINTS	0.30 KB	Delete
POSCAR		Si_Fd-3m.POSCAR.vasp	0.30 KB	Delete
Ouput (OUTC	BANDSTRUCTURE	vasprun.Si_Fd-3m.BANDSTRUCTURE.xml	267.49 KB	Delete
Ouput (OUTC	DFT	vasprun.Si_Fd-3m.DFT.xml	496.22 KB	Delete
Ouput (OUTC	G0W0	vasprun.Si_Fd-3m.GW.xml	499.91 KB	Delete
Ouput (OUTC	NBANDS	vasprun.Si_Fd-3m.NBANDS.xml	154.93 KB	Delete

Figure 4.22: Adding a new material in the ‘Add/Edit Material’ tab.

Figure 4.24 shows how to set the BS-EX bandgap for the calculation with the ID 309. Clicking the 'Save edit' button then writes this change to the database. After that, the data can be seen in the 'Single Materials' tab for ID 309 (Figure 4.25), or in the 'List of Materials' tab if the respective column is chosen from the dropdown menu.



Edit Material

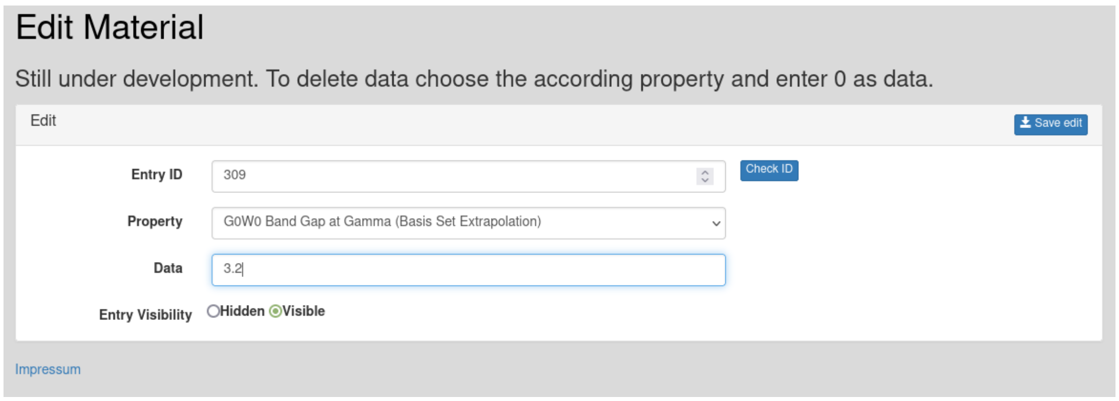
Still under development. To delete data choose the according property and enter 0 as data.

ID found in database: 309 - Si [Fd-3m]

Edit

Entry ID	309
Property	---
Data	
Entry Visibility	Hidden [✓ Visible]

Figure 4.23: Clicking 'Check ID' in the 'Add/Edit Material' tab.



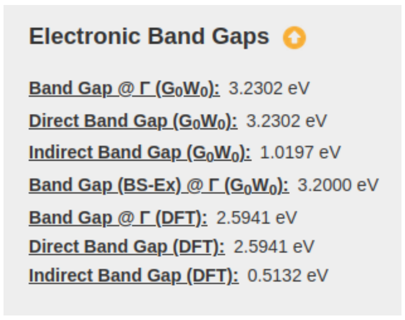
Edit Material

Still under development. To delete data choose the according property and enter 0 as data.

Edit

Entry ID	309
Property	G0W0 Band Gap at Gamma (Basis Set Extrapolation)
Data	3.2
Entry Visibility	Hidden [✓ Visible]

Figure 4.24: Editing a material in the 'Add/Edit Material' tab.



Electronic Band Gaps

Band Gap @ Γ (G_0W_0):	3.2302 eV
Direct Band Gap (G_0W_0):	3.2302 eV
Indirect Band Gap (G_0W_0):	1.0197 eV
Band Gap (BS-Ex) @ Γ (G_0W_0):	3.2000 eV
Band Gap @ Γ (DFT):	2.5941 eV
Direct Band Gap (DFT):	2.5941 eV
Indirect Band Gap (DFT):	0.5132 eV

Figure 4.25: The bandgaps in the 'Single Material' tab after having edited the entry by setting the basis set extrapolation bandgap at the Γ k-point to 3.2 eV.

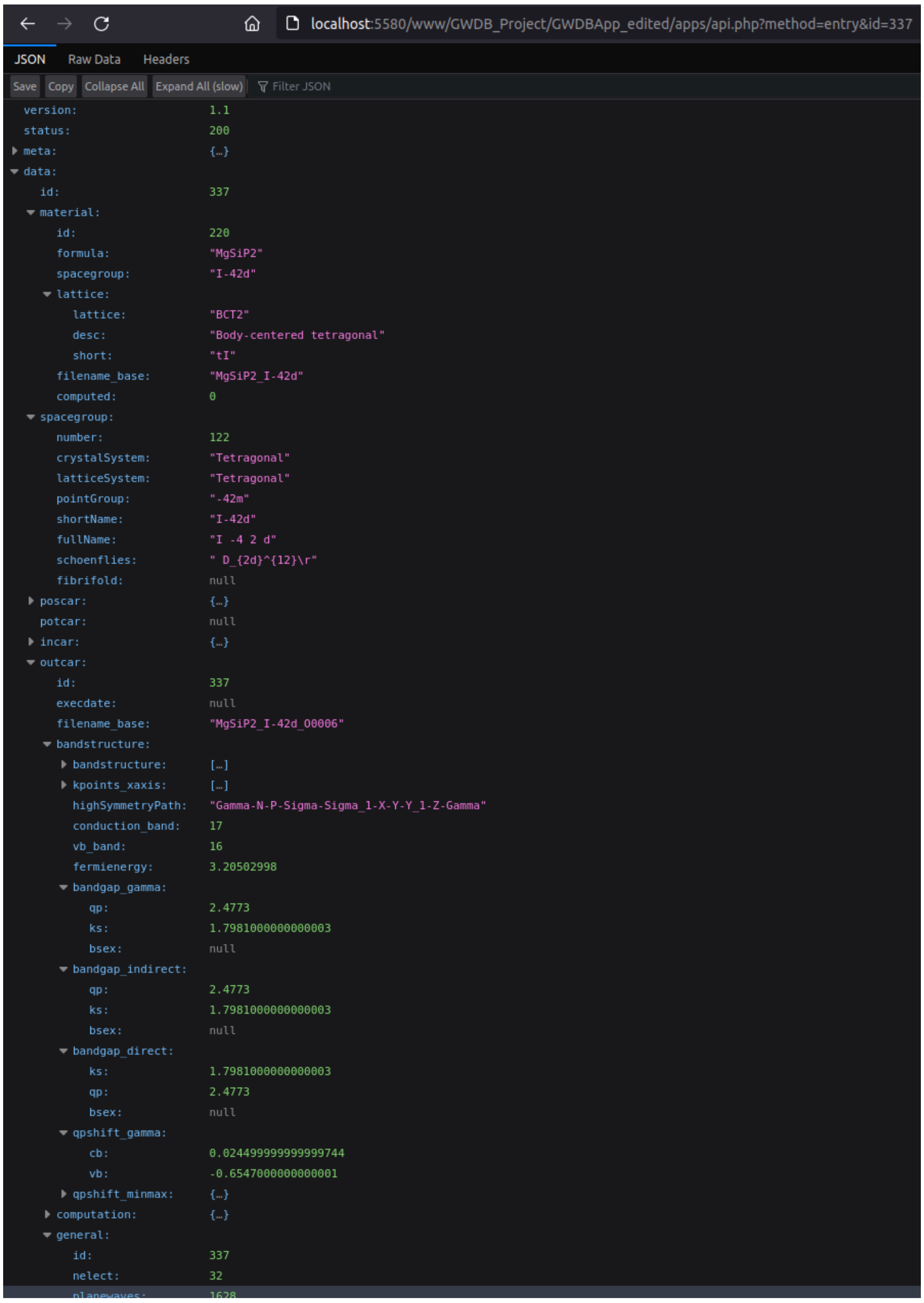
The HTML for the website is written in the `apps` package. The JavaScript code that is responsible for rendering the data and creating the plots on the page can be found in the `js` package and the `modules/js` package. In the JavaScript code, calls are made to api.php via POST requests.

4.3.4 API

The API is the only interface between frontend and backend. It is implemented in server-side PHP code in the file api.php and can thus be technically categorized as part of the backend. Nevertheless, as the connection between frontend and backend and also the only part of the backend that can be accessed directly by users, it is a special case, and will be described here in more detail.

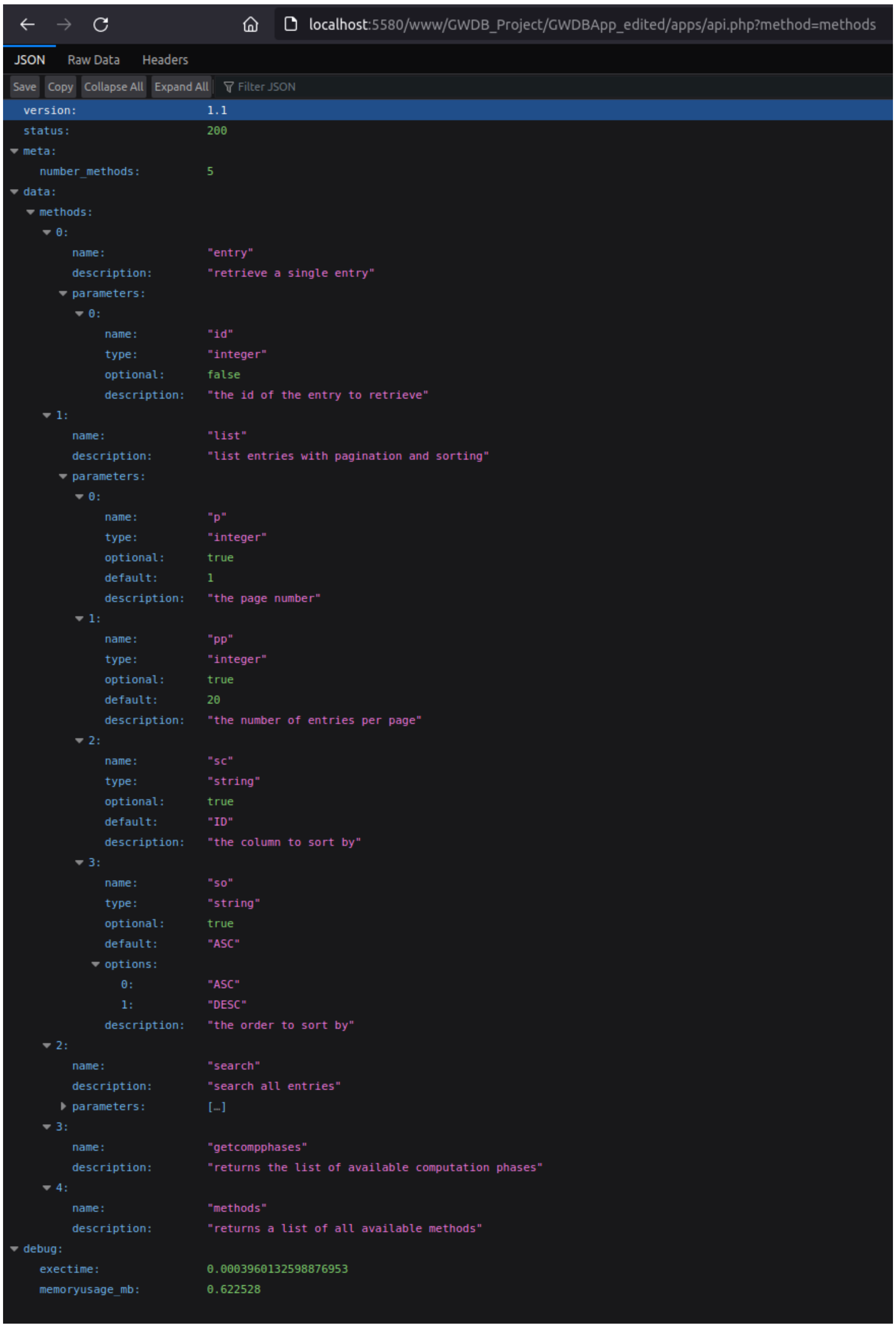
All the data presented to the user by the frontend has been sent to the frontend through the API in the format of a JSON. This makes it easy to use the same backend and the same API to provide the data to arbitrarily many frontends. It also means that users can access all available information without using any frontend at all, by querying the api directly. An example of this is shown in Figure 4.26. The URL for the API-request is the same as the URL to the website, but with 'gwdb.php' replaced by 'api.php', and after that, the request parameters.

It is possible to list and search data using API calls, and with the API-method 'methods' it is possible to list the names and parameters of all available API-methods. The current such list is shown in Figure 4.27. When searching data, the same syntax as in the search bar can be used. For logical AND-search, the & symbol cannot be used since this is the symbol that separates query parameters. Instead, space can simply be used. Figure 4.28 shows the same search as Figure 4.12, conducted via API call. The result is the same entry both times.


localhost:5580/www/GWDB_Project/GWDBApp_edited/apps/api.php?method=entry&id=337

```
version:                    1.1
status:                     200
meta:                       {…}
data:
  id:                       337
  material:
    id:                     220
    formula:                "MgSiP2"
    spacegroup:             "I-42d"
    lattice:
      lattice:              "BCT2"
      desc:                 "Body-centered tetragonal"
      short:                "tI"
    filename_base:          "MgSiP2_I-42d"
    computed:               0
  spacegroup:
    number:                 122
    crystalSystem:          "Tetragonal"
    latticeSystem:          "Tetragonal"
    pointGroup:             "-42m"
    shortName:              "I-42d"
    fullName:               "I -4 2 d"
    schoenflies:            " D_{2d}^{12}\r"
    fibrifold:              null
  poscar:                   {…}
  potcar:                   null
  incar:                    {…}
  outcar:
    id:                     337
    execdate:               null
    filename_base:          "MgSiP2_I-42d_00006"
    bandstructure:
      bandstructure:        […]
      kpoints_xaxis:        […]
      highSymmetryPath:     "Gamma-N-P-Sigma-Sigma_1-X-Y-Y_1-Z-Gamma"
      conduction_band:      17
      vb_band:              16
      fermienergy:          3.20502998
      bandgap_gamma:
        qp:                 2.4773
        ks:                 1.7981000000000003
        bsex:               null
      bandgap_indirect:
        qp:                 2.4773
        ks:                 1.7981000000000003
        bsex:               null
      bandgap_direct:
        ks:                 1.7981000000000003
        qp:                 2.4773
        bsex:               null
      qpshift_gamma:
        cb:                 0.024499999999999744
        vb:                 -0.6547000000000001
      qpshift_minmax:       {…}
    computation:            {…}
    general:
      id:                   337
      nelect:               32
      planewaves:           1628
```

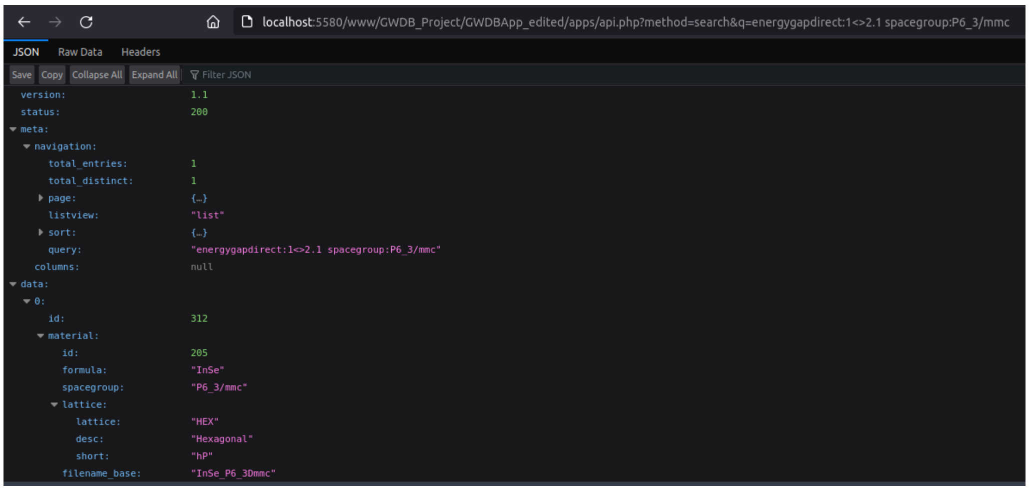
Figure 4.26: An API-Request in the Firefox browser with the url '[...]apps/api.php?method=entry&id=337'. The result is a JSON with all available information to the entry with the if 337 in the GW-Database.



```
localhost:5580/www/GWDB_Project/GWDBApp_edited/apps/api.php?method=methods
```

```
  version:                    1.1
  status:                     200
▾ meta:
    number_methods:           5
▾ data:
  ▾ methods:
    ▾ 0:
        name:                 "entry"
        description:          "retrieve a single entry"
      ▾ parameters:
        ▾ 0:
            name:             "id"
            type:             "integer"
            optional:         false
            description:      "the id of the entry to retrieve"
    ▾ 1:
        name:                 "list"
        description:          "list entries with pagination and sorting"
      ▾ parameters:
        ▾ 0:
            name:             "p"
            type:             "integer"
            optional:         true
            default:          1
            description:      "the page number"
        ▾ 1:
            name:             "pp"
            type:             "integer"
            optional:         true
            default:          20
            description:      "the number of entries per page"
        ▾ 2:
            name:             "sc"
            type:             "string"
            optional:         true
            default:          "ID"
            description:      "the column to sort by"
        ▾ 3:
            name:             "so"
            type:             "string"
            optional:         true
            default:          "ASC"
          ▾ options:
              0:              "ASC"
              1:              "DESC"
            description:      "the order to sort by"
    ▾ 2:
        name:                 "search"
        description:          "search all entries"
      ▸ parameters:           […]
    ▾ 3:
        name:                 "getcompphases"
        description:          "returns the list of available computation phases"
    ▾ 4:
        name:                 "methods"
        description:          "returns a list of all available methods"
▾ debug:
    exectime:                 0.0003960132598876953
    memoryusage_mb:           0.622528
```

Figure 4.27: An API-Request in the Firefox browser with the url '[...]apps/api.php?method=methods'. The result is a JSON with all available API-methods and their parameters. The available API-methods are 'entry', 'list', 'search', 'getcompphases', and 'methods'.


localhost:5580/www/GWDB_Project/GWDBApp_edited/apps/api.php?method=search&q=energygapdirect:1<>2.1 spacegroup:P6_3/mmc
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON

```
version:                    1.1
status:                     200
▼ meta:
  ▼ navigation:
      total_entries:        1
      total_distinct:       1
    ▶ page:                 {…}
      listview:             "list"
    ▶ sort:                 {…}
      query:                "energygapdirect:1<>2.1 spacegroup:P6_3/mmc"
    columns:                null
▼ data:
  ▼ 0:
      id:                   312
    ▼ material:
        id:                 205
        formula:            "InSe"
        spacegroup:         "P6_3/mmc"
      ▼ lattice:
          lattice:          "HEX"
          desc:             "Hexagonal"
          short:            "hP"
        filename_base:      "InSe_P6_3Dmmc"
```

Figure 4.28: An API-Request in the Firefox browser with the url '[...]apps/api.php?method=search&q=energygapdirect:1<>2.1 spacegroup:P6_3/mmc'. The result is a JSON containing all entries that fit the search criteria.

4.4 Adaptations to the GW-Database

While the GW-Database was created in 2016, there were still some necessary additions and improvements to be made before it could be deployed. First and foremost, this concerned the bandgaps and the bandstructure plot, which were missing some crucial data. These were not the only improvements that were made, however. During the course of this thesis there were a number of changes and improvements made to the GW-Database, and an overview of them will be given here.

4.4.1 Bandgaps and Bandstructure

Originally, only the indirect bandgap and the bandgap at Γ were calculated and saved in the database, and the indirect bandgap was simply called 'Energy Gap', without specifying further. Now, the direct bandgap has been added to the code and to the database for both DFT and G_0W_0, and the names have been clarified. The direct bandgap was also added to the keyword search. The bandgap at Γ was also being calculated in a problematic way: In the code, it was assumed that Γ would always be the first and last k-point in the KPOINTS file. This was true for all the data that was uploaded to the database at the time, but is not true in general, as the high-symmetry k-points in the KPOINTS file for the bandstructure calculation can be listed in any order. Now, the bandgap at the Γ k-point is being calculated in a more general way that works for any KPOINTS file. Additionally, only the BS-EX bandgap at Γ was implemented. Now, the direct and indirect basis set extrapolation bandgaps, too, can be saved in the database, edited, and shown on the website.

Some improvements were also made the quasi-particle shift, which was always parsed for the first k-point that happened to be written in the KPOINTS file for the bandstructure. This is not well-defined, however, since the quasi-particle shift is dependent on k. Now, the quasi-particle shift is calculated and saved at the Γ k-point for both the valence band and the conduction band, and additionally at the minimum of the conduction band and at the maximum of the valence band. This way it is possible to get a measure for how much the bandgap could be affected by the quasi-particle shift.

A lot of work was done on the bandstructure plot. It did not include labels for the x-axis, which should be labelled with high-symmetry k-points. These had to be parsed from the input files, translated from 3D coordinates into letters, and plotted correctly. The translation from coordinates to letters was challenging, as extensive information about the lattice is necessary to calculate the letters from the coordinates, and the code had to be restructured to facilitate this functionality. The letters of the k-points path are now saved in the database in the `outcar_mod_bandstructure` table, and the frontend JavaScript code can now use this information to label the x-axis of the bandstructure plot correctly.

It was previously also not possible to show a bandstructure with discontinuities, meaning a bandstructure which was calculated with a KPOINTS file with k-points that do not follow a continuous path. To remedy this, the bandstructure is now treated as an array of plotting data, with each element of this array being a continuous part of the plot. Using this format, the bandstructure can be plotted correctly on the frontend. An example of such a bandstructure plot is shown in Figure 4.29.

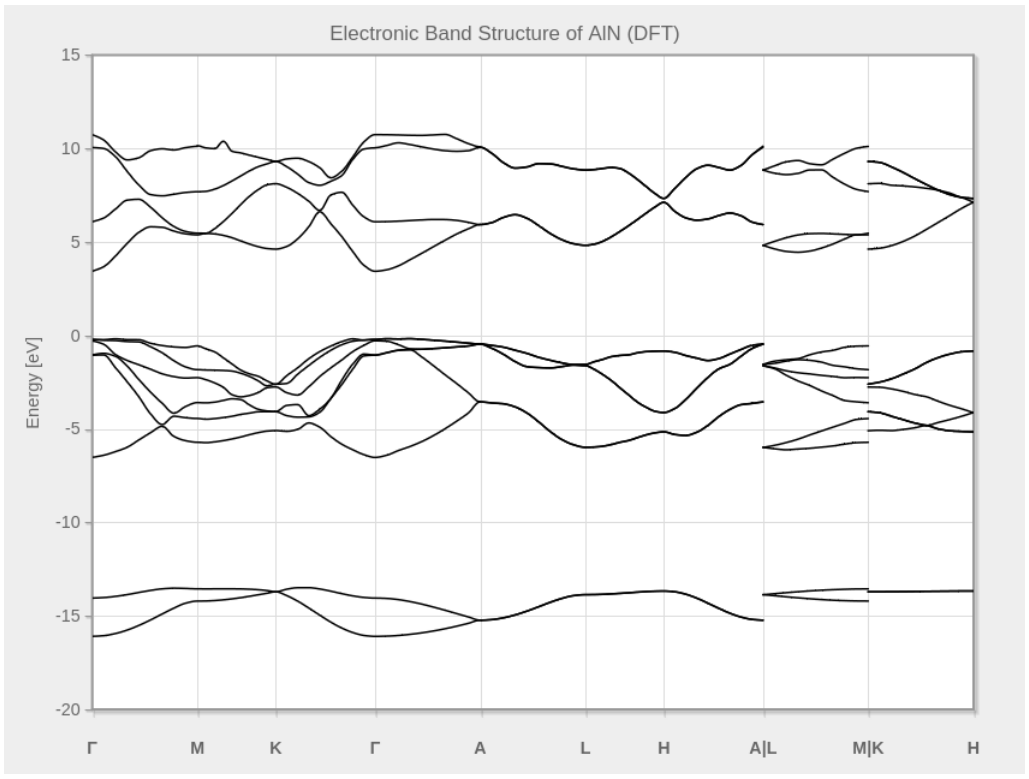


Figure 4.29: A bandstructure plot of AlN where the high-symmetry k-points path is not continuous.

4.4.2 Presentation

The 'Single Material' tab in general was restructured and some data was renamed in order to make the page more clearly arranged, and to make the naming scheme more consistent. For the IMDT plot, the optical conductivity was added as shown in Figure 4.15. The optical conductivity is not saved in the database, but calculated from the relative permittivity by the JavaScript code. The formula of the material was also not shown correctly in the title of the plot, which is fixed now. An impressum was also added to the website.

In the 'List of Materials' tab, some small usability improvements were implemented.

For instance, a ‘Clear Search’ button was added, because it was otherwise sometimes frustrating to get back to all results after having used the search function. Another thing that was changed was that a search that does not yield any results now returns and empty list. Before, the user was instead shown a ‘Error processing the request.’ error banner, which may be confusing and frustrating to users, as it is not clear that the search actually did work and there are just no entries that fit the search criteria. When choosing which columns to show from the ‘Choose Columns’ dropdown menu, the page used to reset to page 1 every time. Now, the page does not change when columns are added or removed.

In the ‘Add/Edit Material’ tab, there is now a list of files on the page that should be uploaded, as shown in Figure 4.21. Without this, it is difficult to know for the user which files are expected by the website. It should be noted here that, since only authorized persons should be able to upload or edit files, it is not necessary to write a longer explanation or instructions here, as the uploader should already be familiar with the basic procedure and can ask for help if needed. A short list of expected files is nevertheless useful, to see at a glance what files are needed.

4.4.3 Added Functionality

A new feature to hide specific entries was implemented, accessible in the ‘Edit Material’ tab, as described in Section 4.3.3 . For this feature, the boolean column ‘hidden’ has been added to the `materials_list` table. Every database query in the PHP code has been adapted to only show entries where ‘hidden’ has the value of false. In the ‘Edit Material’ section in the ‘Add/Edit Material’ tab, an entry can be set to be hidden or visible. This is a highly useful new feature, because without it, there would be no way to delete or hide anything that was once uploaded to the database except using manual SQL queries. That would be difficult and error prone, because of the interconnected nature of the data that is distributed among the different tables.

In the ‘PSE’ tab, an Update button was added. Clicking this button causes the periodic table graphic to be updated with all the materials that are currently in the database. This functionality was already programmed, but was not used anywhere, and it took some time to find it. Now, whenever new materials are added to the GW-Database, it is easy to update the graphic to reflect that.

There was also an important security improvement. While the ‘Add/Edit Material’ tab is behind a rudimentary password protection, as seen in Figure 4.20, this does not apply to API-calls. It is probable that this was not noticed by the creator of the password protection, and thus no steps were taken to prevent unauthorized access over direct API-calls. This is particularly dangerous because the 'Edit Material' section theoretically allows the changing of any property in the `outcar_mod_bandstructure` table, and this would be very easy to do with an API call. Steps have now been taken to prevent API calls from outside this frontend and without knowledge of the password from updating or adding entries. However, it must be reiterated that someone more experienced in cybersecurity is needed to ensure that this is really secure.

4.4.4 Bug Fixes

Improvements have also been made to the file uploading functionality, such as better error handling and alerting, as well as a bug fix concerning the 'Delete' buttons for the individual files. In the 'Edit Material' section, a bug was fixed concerning the 'Check ID' functionality, which used to check for a the ID of a material instead of the ID of an entry.

One big quality-of-life improvement for uploading of files is a bug fix concerning the session key. Previously, whenever a new entry was added, the password had to be re-entered afterwards. The reason for this was that the ID of the added entry was appended to the URL. The session now ignores this added ID so that the user remains logged in and does not have to re-enter the password.

In the `const_spacegroups` table, each spacegroup is mapped to its crystal system. However, in order to parse the lattice information from the POSCAR and calculate the high-symmetry k-points, the PHP coded uses the lattice system, not the crystal system. This led to a situation where rhombohedral lattices could not be parsed correctly. This has been fixed by adding the lattice system to the `const_spacegroups` table. While working on this problem, it was discovered that there were numerous instances where lattice geometry was processed incorrectly because degrees were used with PHP's sine and cosine functions, which take radians as their argument. These bugs have also been fixed.

When making an API call with the method 'methods', there was a bug where only one parameter was shown for each method, because every time a new parameter was added, it overwrote the previous one. Now, all parameters are correctly added and communicated.

4.4.5 Behind the scenes

One of the biggest challenges of working with the GW-Database was the lack of documentation or comments. A lot of time was spent trying to understand what a particular function does, or what datatype a certain variable is. Since PHP is a dynamically typed language, the data types of variables are not declared, and since it is a weakly typed language, the datatype of a variable can change at any time. It should be clear that this makes it difficult to understand uncommented code, especially since in object-oriented programming, the datatype of a variable can be any of a large number of classes.

This is why efforts have been made to make it easier to understand the GW-Database for future programmers. Specifically, every time a method or class or even a particularly tricky variable became understood, a comment was added to explain it for future readers. The PHPDoc[49] format was used extensively to describe methods and classes, and a particular focus was put on documenting the data types/classes of the parameters of methods and field variables of classes. Any methods that were added from scratch in the course of this project were documented using PHPDoc comments, and type declarations were added to their parameters. Additionally, some code has been refactored with a focus on readability.

In this way, a majority of the important methods in the backend now have PHPDoc documentation, and the parts of the code that were most challenging to understand have been either refactored or received explanatory comments. While these changes will not be noticeable to users of the GW-Database, they are expected to dramatically improve the readability of the code and the maintainability of the project as a whole.

Concerning code maintainability, an autoloading functionality has also been implemented, to avoid having to import every class by hand. This is especially important when classes are moved between packages, as without autoloading, this would necessitate editing every import statement of this class in the entire code. Now, instead, this happens automatically. Some code refactoring was also done, especially in the `classes` package, to bring the code more in line with coding conventions such as having only one class per file.

The MariaDB database was also changed to better align with conventional practices. Previously, even though tables were joined using their ID columns, there were no foreign keys implemented in the database. Not only can this lead to problems with data integrity, it also makes it more difficult to understand the database schema if one is not familiar with it. To remedy this, foreign keys were implemented according to the connections shown in the ER-diagram in Figure 4.3.

It was also made easier to reset the MariaDB database. For development and testing, the database often had to be altered or reset, and incorrect data had to be uploaded to test error handling. It was tedious and time-consuming to restart and reset the database, especially since it was originally provided as a database dump including both the structure of the database and the data. To make the process easier, the structure parts of the database dump were extricated to make it possible to create an empty database. The const tables such as `const_spacegroups` or `const_elements` do not take data from the uploaded files and should have all their data from the start, so the SQL statements that insert data into these tables are written into a separate .sql file. As soon as all the structure-relevant SQL statements are executed, this file should be executed to fill the const tables. Every time the structure of the database was altered, e.g. to include the direct bandgaps, the SQL-statements to alter the database were saved in .sql files. Over time, quite a few .sql files are created, and have to be executed in a specific order in order to create an empty database.

To make this process easier, a Bash script was created that automatically resets the database by dropping the existing database and then executing all these .sql files in the correct order. Bash scripts were also created for creating a database dump from the current database, and for restoring the database from a database dump file. The latter is useful to get a database containing data that had been uploaded at an earlier point in time, but it is problematic to do this to restore early database version which were missing some alterations. It is possible to add these alterations to the older version of the database, but it may require knowledge of the exact structure of the database when the database dump file was made, and knowledge of which alterations had already been made and which ones had not. Instructional .txt files have been created to document how to use these Bash scripts, and also how to solve some common errors that might arise.

A lot more smaller fixes and improvements were made to the code and the project, but they are too minor and numerous to list all of them here.

4.5 Data Processing Example

Having completed a calculation as described in Section 4.2, the input and output files can be uploaded to be GW-Database. As soon as they have been uploaded, the data is parsed by the application, and can then be retrieved and viewed by any user. The following is an example of how this process works internally.

As soon as the ‘Save item’ button is clicked in the ‘Add Material’ section, the JavaScript method `saveNewMaterial()` is called in the class `GWDBAddMaterial`. This method calls the method `getPayload()`, which takes all the information from the input fields (the spacegroup and formula that the user typed in, as well as the files that were uploaded) and returns it. `saveNewMaterial()` then checks this information to see if it is correct, and shows an error to the user if not. If it is correct, the method `PageBuilder.processAJAXHandler()` is called with this data and the method name that tells api.php to add this data to the database. `PageBuilder.processAJAXHandler()` sends the data to the API URL via a POST request, similar to the API requests that were described in Section 4.3.4, and handles any errors that might occur.

In api.php, the method name is extracted and the corresponding function is executed. `AddMaterial::initWithPayload()` is called in PHP, with the formula, spacegroup, and files as parameter. It transforms this JSON parameter into PHP variables. In the `importMaterialAndResults()` method, different extractor methods are called which parse the information from the uploaded files and save them as instances of their respective PHP classes such as `Poscar` and `Material`. When all the data has been parsed, these objects are combined into one `GWEntry`. This object, which contains all the information, is then saved in the database. The `save()` method of the `GWEntry` class calls the `saveSubObject()` method for each of its subobjects. In this way, all the information is saved to the correct tables. The database IDs of the subobjects are saved in the `GWEntry` instance, and are saved in the `materials_list` table at the very end.

It should be noted that a large part of the backend code in the `classes` package is dedicated to parsing information from the files and transforming it into the desired format before saving it in the database. For instance, the bandstructure has to be extracted in such a way that makes it easy to plot later, along with information about the bandgaps and quasi-particle shifts. The geometry from the POSCAR file has to be parsed and structural information about the Bravais lattice and unit cell has to be calculated from it. Parsing the POSCAR has the additional challenge of it being manually created, and therefore not necessarily following certain conventions, such as the order of the lattice vectors. For these reasons, even though the above description of 'The information is parsed and saved' sounds simple, a large portion of the code was written for this task.

Once the data has been added to the database, api.php replies with a success message and the ID of the newly added entry. The frontend which made the API call receives this reply and can now show the user this success message and ID. If the user now clicks on the ID or looks at the list of materials, the new entry is called from the database and shown to the user, just like any other entry:

Clicking on the ID of an entry calls the javascript method `PageBuilder.getEntry()` with the ID as a parameter. This method checks if this ID is found in the cache, and if it is, it simply gets the information from the cache. If it is not in the cache, the method makes an API-call with the method-name 'entry' and the ID as parameters, and the API responds with a JSON containing all the information about the entry with this ID (similar to the API call shown in Figure 4.26). This JSON is then stored in the cache. In either case, the `PageBuilder.setEntry()` method then uses the information from this JSON to populate all the HTML fields in the 'Single Material' tab and shows it to the user.

To get the data from the database, api.php calls the method `getEntry()` in the `GWEntryDatabase` class. This method first gets the `poscar_id`, `incar_id`, `material_id`, and `outcar_id` connected to the given ID in the `materials_list` table, and then uses these IDs to fetch the information from the respective `poscar`, `incar`, `materials`, and `outcar` tables. This information is transformed into an instance of the `GWEntry` class. The API then calls `GWEntry`'s method `convertObjectToStdClass()`, which transforms the `GWEntry` instance into the JSON data that is then returned to the frontend.

Chapter 5

Conclusion

5.1 Summary

Databases providing results of ab-initio simulations play a crucial role in computational materials science, offering researchers access to well-organized and analyzable data that can serve as a reference for comparison, or as a training set for machine learning algorithms. While open-access databases focused on DFT simulation data abound, DFT simulations have limitations in predicting certain material properties accurately. The GW approximation is more accurate than DFT, at the cost of being more computationally expensive. This makes GW data a valuable resource as an accurate reference.

This thesis introduces a database for sharing and accessing GW data. It enables authorized users to upload VASP input and output files of well-converged calculations. These files are subsequently parsed, processed, and integrated into the GW-Database, allowing all other researchers to access this high-quality data either on the website or using API calls.

Simulation results for a variety of semiconducting and insulating materials have already been added to the database, and more can be added at any time. The process of how to run VASP calculations to produce GW data for the GW-Database is described in this thesis. It involves performing DFT calculations followed by a GW calculation that uses the DFT results as a starting point. The input and output files of these VASP calculations can be added to the database by uploading them on the website. The end result is a searchable database for all uploaded calculations, providing a summary of important properties such as bandgaps, as well as all uploaded input and output files.

The MariaDB database, PHP backend, and JavaScript/HTML frontend making up the GW-Database are described in detail in this thesis, providing helpful documentation for users and future maintainers of the database. The GW-Database uses a Model-View-Controller pattern, ensuring an efficient and scalable design. Moreover, the GW-Database was enhanced by adding information about more material properties, particularly the smallest direct bandgap. Other features that were added include a more useful plot of the bandstructure, manual data input for bandgaps calculated using the basis-set approximation, the ability to update the material graph, and the hiding of entries. Comprehensive PHPDoc comments were added to the code, addressing the previous lack of documentation.

With these advancements, the GW-Database now stands fully equipped as a functional and invaluable resource for materials researchers seeking access to GW data. The availability of accurate reference data will undoubtedly propel advancements in materials research, benefiting researchers across various domains relying on accurate material property information.

5.2 Perspective

While the GW-Database now provides all essential functionalities, there remains potential for further improvement. Due to time constraints, it was necessary to prioritize certain features over others, focusing on those that promised the greatest benefit relative to the invested effort. The following section presents recommendations for potential future improvements to the GW-Database.

As was already implied in Section 4.4, several bugs and other difficulties were encountered in the methods which extract information from the uploaded files and transform it into different formats. A lot of these problems could be avoided by using an external library. For instance, Pymatgen (Python Materials Genomics)[50] is an open-source Python library with the ability to parse VASP files into custom python classes, and a variety of methods to interact with instances of these classes, and transform or extract information. It would require not insignificant code refactoring, but it should be possible to integrate this library, or one similar to it, into the GW-Database. Using an externally developed, maintained, and tested library for the data parsing would significantly reduce the likelihood of bugs occurring. It would also greatly simplify the future addition of new features.

If the GW-Database becomes more popular, with more people being aware of and even relying on it, the security of the application should be increased. As mentioned previously, neither the password protection nor the API protection that prevents unauthorized persons from changing the data within the database was created by someone with a cybersecurity background. Without expertise and experience, it is difficult to judge how protected or vulnerable the application really is. Therefore, it would be beneficial to have a person with some experience in cybersecurity look over the GW-Database and, if necessary, improve the security features.

To improve the API querying capabilities of the GW-Database, the API specification developed by the Open Databases Integration for Materials Design (OPTIMADE)[51] consortium could be implemented. This API specification was developed with the aim of simplifying the querying of different databases, which would otherwise require the knowledge of very different querying syntaxes. Several popular computational materials databases already support the OPTIMADE API standard. Implementing this standard would allow researcher familiar with OPTIMADE to easily query the GW-Database.

Currently it is possible to add calculations that are not fully converged, have partial occupancy, or have similar undesirable properties to the database. It would be useful to check for these problems when the user clicks the ‘Save Item’ button, show a warning message about the problem, and ask the user if they are sure they want to add this calculation to the database. This would prevent accidentally adding data from problematic calculations. The reason implementing this feature has not been a priority is that only authorized users are able to upload calculations, and should be familiar enough with the GW-Database to be able to avoid making this mistake. If any user of the website were able to add data, this feature would be far more important. However, it would still be a quality of life improvement to add this check, since accidents can nevertheless happen to anyone. Incorporating a library like the aforementioned Pymatgen could help with this issue as well, as it such checks are already implemented there.

When editing data in the ‘Edit Material’ section, it would be useful to save the metadata of each edit in the database. Saving the previous value, new value, who changed it, and the date of the change, would make it possible to keep track of all changes that are made. As more changes are made, this would create a useful way of monitoring them.

Another change in the ‘Edit Material’ section could be providing a search functionality and list of entries, similar to the ‘List of Materials’ tab. That would make it easier to find the entry one wants to change, and it would no longer be necessary to find and remember the ID of the entry before going to the ‘Add/Edit Material’ tab.

A graphical user interface (GUI) for searching would improve usability of the GW-Database. As Section 4.3.3 shows, the search syntax is not always immediately intuitive. A GUI may not offer more functionality than the current search syntax, but it would be clearer to users. Another improvement of the search would be to implement a way of using a logical OR in the search.

For overall usability, it may make the frontend more appealing if the style of the website were changed to a more modern design.

Overall, while some aspects of the GW-Database could still be improved, it provides a simple and easy to use way to share accurate GW data between researchers.

Acknowledgements

I extend my sincere appreciation to my thesis supervisor Univ.-Prof. Dr. techn. Cesare Franchini for all his help and guidance, not only with the thesis itself but also with all the organizational aspects surrounding it. He was always ready to give advice whenever I had any questions, and helped me overcome any roadblocks I might face.

I also wish to acknowledge and thank Karl Prikopa for his work in creating the GW-Database. His contributions are foundational to this project.

I could never have gotten this far without the support of my friends and family, who were always there to help me through difficult times. In this context, special thanks must go to to Doris Pötzelberger and Volker Pötzelberger, for their unwavering support over the last couple of decades and for always encouraging my curiosity for science and technology, and to Gernot Pötzelberger for innumerable though-provoking discussions about mathematics, physics, and technology.

I am also grateful to all of the members of the Quantum Materials Modelling Group. When I started working on this project, all colleagues here was extremely welcoming and nice, and it makes me glad to have become friends with them. They made the office a welcoming work environment and always provided moral support and helpful advice.

Last but not least, I want to especially thank my colleague and friend Florian Ellinger, who was always ready to answer technical questions, and to share with me his vast expertise in the topics of ab-initio calculations and VASP. Our discussions helped me gain valuable insights into the topic that would have been otherwise very difficult to get. Florian also has my eternal gratitude for proof-reading the thesis and giving me detailed and invaluable feedback. Thank you, Flo. This thesis would be much worse without you.

Bibliography

[1] M. J. Mehl et al. “The AFLOW Library of Crystallographic Prototypes: Part 1”. In: *Computational Materials Science* 136 (Aug. 1, 2017), S1–S828. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2017.01.017.

[2] A. Jain et al. “Commentary: The Materials Project: A Materials Genome Approach to Accelerating Materials Innovation”. In: *APL Materials* 1.1 (July 18, 2013), p. 011002. ISSN: 2166-532X. DOI: 10.1063/1.4812323.

[3] *Materials Genome Initiative | WWW.MGI.GOV*. URL: https://www.mgi.gov/ (visited on 04/27/2023).

[4] P. Hohenberg and W. Kohn. “Inhomogeneous Electron Gas”. In: *Physical Review* 136 (3B Nov. 9, 1964), B864–B871. DOI: 10.1103/PhysRev.136.B864.

[5] *Calculation Details*. URL: https://docs.materialsproject.org/methodology/materials-methodology/calculation-details (visited on 05/03/2023).

[6] A. Jain et al. “A High-Throughput Infrastructure for Density Functional Theory Calculations”. In: *Computational Materials Science* 50.8 (June 1, 2011), pp. 2295–2310. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2011.02.023.

[7] S. Curtarolo et al. “AFLOW: An Automatic Framework for High-Throughput Materials Discovery”. In: *Computational Materials Science* 58 (June 1, 2012), pp. 218–226. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2012.02.005.

[8] C. Draxl and M. Scheffler. “NOMAD: The FAIR Concept for Big Data-Driven Materials Science”. In: *MRS Bulletin* 43.9 (Sept. 1, 2018), pp. 676–682. ISSN: 1938-1425. DOI: 10.1557/mrs.2018.208.

[9] J. E. Saal et al. “Materials Design and Discovery with High-Throughput Density Functional Theory: The Open Quantum Materials Database (OQMD)”. In: *JOM* 65.11 (Nov. 1, 2013), pp. 1501–1509. ISSN: 1543-1851. DOI: 10.1007/s11837-013-0755-4.

[10] S. Kirklin et al. "The Open Quantum Materials Database (OQMD): Assessing the Accuracy of DFT Formation Energies". In: *npj Computational Materials* 1.1 (1 Dec. 11, 2015), pp. 1–15. ISSN: 2057-3960. DOI: 10.1038/npjcompumats.2015.10.

[11] R. W. Godby, M. Schlüter, and L. J. Sham. "Accurate Exchange-Correlation Potential for Silicon and Its Discontinuity on Addition of an Electron". In: *Physical Review Letters* 56.22 (June 2, 1986), pp. 2415–2418. DOI: 10.1103/PhysRevLett.56.2415.

[12] L. J. Sham and M. Schlüter. "Density-Functional Theory of the Energy Gap". In: *Physical Review Letters* 51.20 (Nov. 14, 1983), pp. 1888–1891. DOI: 10.1103/PhysRevLett.51.1888.

[13] F. Aryasetiawan and O. Gunnarsson. "The GW Method". In: *Reports on Progress in Physics* 61.3 (Mar. 1998), p. 237. ISSN: 0034-4885. DOI: 10.1088/0034-4885/61/3/002.

[14] F. Giustino. *Materials Modelling Using Density Functional Theory: Properties and Predictions*. Oxford University Press, 2014. ISBN: 978-0-19-966243-2.

[15] M. Orio, D. A. Pantazis, and F. Neese. "Density Functional Theory". In: *Photosynthesis Research* 102.2 (Dec. 1, 2009), pp. 443–453. ISSN: 1573-5079. DOI: 10.1007/s11120-009-9404-8.

[16] W. Koch and M. C. Holthausen. *A Chemist's Guide to Density Functional Theory*. John Wiley & Sons, Nov. 18, 2015. 534 pp. ISBN: 978-3-527-80281-4. Google Books: qm5cCwAAQBAJ.

[17] L. Reining. "The GW Approximation: Content, Successes and Limitations". In: *WIREs Computational Molecular Science* 8.3 (2018), e1344. ISSN: 1759-0884. DOI: 10.1002/wcms.1344.

[18] J. P. Perdew and M. Levy. "Physical Content of the Exact Kohn-Sham Orbital Energies: Band Gaps and Derivative Discontinuities". In: *Physical Review Letters* 51.20 (Nov. 14, 1983), pp. 1884–1887. DOI: 10.1103/PhysRevLett.51.1884.

[19] L. Hedin. "New Method for Calculating the One-Particle Green's Function with Application to the Electron-Gas Problem". In: *Physical Review series I* 139.3A (3A 1965), pp. 796–823. ISSN: 0031-899X.

[20] M. S. Hybertsen and S. G. Louie. "Electron Correlation in Semiconductors and Insulators: Band Gaps and Quasiparticle Energies". In: *Physical Review B* 34.8 (Oct. 15, 1986), pp. 5390–5413. DOI: 10.1103/PhysRevB.34.5390.

[21] *GW Approximation of Hedin's Equations - Vaspwiki*. URL: https://www.vasp.at/wiki/index.php/GW_approximation_of_Hedin%27s_equations (visited on 09/15/2023).

[22] F. Ellinger. "High-Throughput GW Calculations". MA thesis. Vienna: Universität Wien, 2020.

[23] M. S. Hybertsen and S. G. Louie. "First-Principles Theory of Quasiparticles: Calculation of Band Gaps in Semiconductors and Insulators". In: *Physical Review Letters* 55.13 (Sept. 23, 1985), pp. 1418–1421. DOI: 10.1103/PhysRevLett.55.1418.

[24] S. E. Gant et al. "Optimally Tuned Starting Point for Single-Shot GW Calculations of Solids". In: *Physical Review Materials* 6.5 (May 16, 2022), p. 053802. DOI: 10.1103/PhysRevMaterials.6.053802.

[25] J. Klimeš, M. Kaltak, and G. Kresse. "Predictive GW Calculations Using Plane Waves and Pseudopotentials". In: *Physical Review B* 90.7 (Aug. 14, 2014), p. 075125. DOI: 10.1103/PhysRevB.90.075125.

[26] S. Körbel et al. "Benchmark Many-Body GW and Bethe–Salpeter Calculations for Small Transition Metal Molecules". In: *Journal of Chemical Theory and Computation* 10.9 (Sept. 9, 2014), pp. 3934–3943. ISSN: 1549-9618. DOI: 10.1021/ct5003658.

[27] X. Blase, C. Attaccalite, and V. Olevano. "First-Principles GW Calculations for Fullerenes, Porphyrins, Phtalocyanine, and Other Molecules of Interest for Organic Photovoltaic Applications". In: *Physical Review B* 83.11 (Mar. 4, 2011), p. 115103. DOI: 10.1103/PhysRevB.83.115103.

[28] F. Bruneval and M. A. L. Marques. "Benchmarking the Starting Points of the GW Approximation for Molecules". In: *Journal of Chemical Theory and Computation* 9.1 (Jan. 8, 2013), pp. 324–329. ISSN: 1549-9618. DOI: 10.1021/ct300835h.

[29] A. Silberschatz, H. F. Korth, and S. Sudarshan. *Database System Concepts*. 6th ed. New York: McGraw-Hill, 2011. 1349 pp. ISBN: 978-0-07-128959-7.

[30] E. F. Codd. "A Relational Model of Data for Large Shared Data Banks". In: *Communications of the ACM* 13.6 (June 1, 1970), pp. 377–387. ISSN: 0001-0782. DOI: 10.1145/362384.362685.

[31] *About MariaDB Server*. MariaDB.org. URL: https://mariadb.org/about/ (visited on 08/04/2023).

[32] *MySQL*. URL: https://www.mysql.com/ (visited on 07/02/2023).

[33] E. Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Deutschland GmbH, 1995.

[34] G. Kresse and J. Hafner. “Ab Initio Molecular Dynamics for Liquid Metals”. In: *Physical Review B* 47.1 (Jan. 1, 1993), pp. 558–561. DOI: 10.1103/PhysRevB.47.558.

[35] G. Kresse and J. Furthmüller. “Efficient Iterative Schemes for Ab Initio Total-Energy Calculations Using a Plane-Wave Basis Set”. In: *Physical Review B* 54.16 (Oct. 15, 1996), pp. 11169–11186. DOI: 10.1103/PhysRevB.54.11169.

[36] P. E. Blöchl. “Projector Augmented-Wave Method”. In: *Physical Review B* 50.24 (Dec. 15, 1994), pp. 17953–17979. DOI: 10.1103/PhysRevB.50.17953.

[37] G. Kresse and D. Joubert. “From Ultrasoft Pseudopotentials to the Projector Augmented-Wave Method”. In: *Physical Review B* 59.3 (Jan. 15, 1999), pp. 1758–1775. DOI: 10.1103/PhysRevB.59.1758.

[38] J. P. Perdew, K. Burke, and M. Ernzerhof. “Generalized Gradient Approximation Made Simple”. In: *Physical Review Letters* 77.18 (Oct. 28, 1996), pp. 3865–3868. DOI: 10.1103/PhysRevLett.77.3865.

[39] *OUTCAR - Vaspwiki*. URL: https://www.vasp.at/wiki/index.php/OUTCAR (visited on 06/02/2023).

[40] W. Setyawan and S. Curtarolo. “High-Throughput Electronic Band Structure Calculations: Challenges and Tools”. In: *Computational Materials Science* 49.2 (Aug. 1, 2010), pp. 299–312. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2010.05.010.

[41] *VSC: Home*. URL: https://vsc.ac.at/home/ (visited on 05/26/2023).

[42] *PHP: Hypertext Preprocessor*. July 6, 2023. URL: https://www.php.net/index.php (visited on 07/08/2023).

[43] *JavaScript | MDN*. July 8, 2023. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (visited on 07/08/2023).

[44] P. P.-S. Chen. “The Entity-Relationship Model—toward a Unified View of Data”. In: *ACM Transactions on Database Systems* 1.1 (Mar. 1, 1976), pp. 9–36. ISSN: 0362-5915. DOI: 10.1145/320434.320440.

[45] *Draw.Io*. URL: https://www.drawio.com/ (visited on 06/16/2023).

[46] *JSON*. URL: https://www.json.org/json-en.html (visited on 08/17/2023).

[47] *PHP: MySQLi - Manual*. URL: https://www.php.net/manual/en/book.mysqli.php (visited on 06/09/2023).

[48] *Jmol Wiki - JSmol*. URL: https://wiki.jmol.org/index.php/JSmol (visited on 06/23/2023).

[49] *PHPDoc Standard Recommendation*. GitHub. URL: https://github.com/php-fig/fig-standards/blob/master/proposed/phpdoc.md (visited on 07/28/2023).

[50] S. P. Ong et al. "Python Materials Genomics (Pymatgen): A Robust, Open-Source Python Library for Materials Analysis". In: *Computational Materials Science* 68 (Feb. 2013), pp. 314–319. ISSN: 09270256. DOI: 10.1016/j.commatsci.2012.10.028.

[51] C. W. Andersen et al. "OPTIMADE, an API for Exchanging Materials Data". In: *Scientific Data* 8.1 (1 Aug. 12, 2021), p. 217. ISSN: 2052-4463. DOI: 10.1038/s41597-021-00974-z.