GusSiesta Platform: Uma Aplicação Web para Conversão de Arquivos XYZ para FDF com Visualização 3D e Gerenciamento de Pseudopotenciais

Gustavo Garcia Pereira, André Flores dos Santos
Curso de Ciência da Computação
UFN – Universidade Franciscana
Santa Maria – RS
g.pereira@ufn.edu.br, andre.flores@ufn.edu.br

***Resumo*—Este trabalho apresenta o desenvolvimento da GusSiesta Platform, uma aplicação web construída com Django para apoiar pesquisadores e estudantes que utilizam o software de simulação de materiais SIESTA (*Spanish Initiative for Electronic Simulations with Thousands of Atoms*). A plataforma oferece uma interface intuitiva para a conversão de arquivos de coordenadas moleculares em arquivos de entrada do SIESTA, com visualização tridimensional (3D) interativa via 3Dmol.js, configuração de parâmetros de simulação, download de arquivos de pseudopotencial, sistema de autenticação de usuários e containerização com Docker. A plataforma implementa ainda histórico de conversões por usuário e salvamento de configurações de parâmetros, totalizando 88 testes automatizados com 100% de aprovação. A validação foi realizada com a molécula de Heparina, demonstrando que o arquivo FDF (*Flexible Data Format*) gerado atende à especificação do SIESTA. Uma avaliação preliminar da plataforma foi conduzida pela própria equipe de desenvolvimento, sugerindo indícios de que a automação do processo pode reduzir a barreira de entrada para novos usuários da ferramenta, hipótese que ainda carece de confirmação por meio de testes com usuários externos. A plataforma está disponível como código aberto e pode ser implantada por instituições de ensino por meio de contêineres Docker.**

***Palavras-chave*: SIESTA; DFT; aplicação web; Django; pseudopotenciais**

I. Introdução

A simulação computacional de materiais consolidou-se como uma das principais ferramentas metodológicas na física da matéria condensada, na química computacional e na ciência dos materiais [1]. O avanço dos métodos de primeiros princípios permitiu calcular propriedades eletrônicas, estruturais e termodinâmicas de sistemas sem parâmetros empíricos, conferindo maior poder preditivo às conclusões obtidas [2].

Nesse cenário, o SIESTA destaca-se como solução amplamente adotada pela comunidade científica, tratando sistemas com milhares de átomos a partir da Teoria do Funcional da Densidade (DFT, do inglês *Density Functional Theory*) com orbitais atômicos localizados (PAO, do inglês *Pseudo-Atomic Orbitals*) [3].

Apesar de sua relevância, o SIESTA impõe barreira de entrada considerável: a preparação dos arquivos FDF exige conhecimento simultâneo da sintaxe do formato, dos parâmetros físicos da simulação e da disponibilidade de arquivos de pseudopotencial (`.psf`) compatíveis. Esse processo manual é tedioso e propenso a erros, representando obstáculo concreto à adoção do SIESTA em disciplinas de graduação e no início de projetos de pesquisa.

Esse cenário é especialmente relevante na Universidade Franciscana (UFN), em Santa Maria-RS, onde o Laboratório de Simulação Molecular (LASIMON) [4] realiza pesquisas *ab initio* (termo latino que designa simulações realizadas a partir de princípios físicos fundamentais, sem parâmetros ajustados empiricamente) com o SIESTA, criando oportunidade direta para que contribuições de engenharia de software apoiem a produtividade científica.

O **objetivo geral** deste trabalho é desenvolver a GusSiesta Platform, uma aplicação web de código aberto em Django que automatiza a geração de arquivos FDF a partir de arquivos no formato XYZ, reduzindo a barreira de uso do SIESTA para iniciantes. Os objetivos específicos são: (i) implementar o algoritmo de conversão de XYZ para FDF com suporte aos principais parâmetros do SIESTA; (ii) integrar visualizador 3D via 3Dmol.js; (iii) implementar gerenciamento e download de pseudopotenciais; (iv) desenvolver autenticação de usuários; e (v) containerizar a aplicação com Docker.

A principal contribuição deste trabalho é, portanto, disponibilizar o único conjunto de ferramentas, dentre os analisados na Seção III, que integra em ambiente web — sem necessidade de instalação local — a conversão de arquivos, a visualização molecular e o gerenciamento de pseudopotenciais voltados especificamente ao SIESTA. A metodologia adotada para o desenvolvimento seguiu os princípios da metodologia ágil *Extreme Programming* (XP) [5], detalhada na Seção IV, com entregas incrementais validadas em ciclos curtos e cobertura por testes automatizados. O código-fonte está disponível em [6].

O restante deste artigo está organizado da seguinte forma: a Seção II apresenta o referencial teórico sobre o SIESTA, os formatos de arquivo envolvidos e as tecnologias empregadas; a Seção III discute trabalhos relacionados e posiciona a contribuição deste trabalho frente a eles; a Seção IV

detalha a metodologia de desenvolvimento, a arquitetura do sistema e o algoritmo de conversão; a Seção V apresenta os resultados, a validação obtida e as limitações identificadas; e a Seção VI conclui o trabalho.

II. Referencial Teórico

A. O Software SIESTA

O SIESTA é um programa para cálculos de estrutura eletrônica e dinâmica molecular (MD, do inglês *Molecular Dynamics*) *ab initio*, projetado para sistemas em larga escala com custo computacional reduzido [3]. Sua base teórica é a DFT: por meio das equações de Kohn-Sham [7], o problema de múltiplos elétrons interagentes é mapeado em um sistema não interagente movendo-se em um potencial efetivo, reduzindo drasticamente a dimensionalidade matemática do problema.

O principal diferencial do SIESTA reside no uso de orbitais PAO estritamente confinados: ao assumir que os orbitais se anulam além de um raio de corte, as matrizes hamiltonianas tornam-se esparsas, permitindo escalonamento $\mathcal{O}(N)$ linear em vez do cúbico $\mathcal{O}(N^3)$ típico dos métodos tradicionais [3]. Devido a essa eficiência e à sua licença de código aberto, o SIESTA é amplamente adotado para investigação de nanoestruturas, defeitos cristalinos e macromoléculas biológicas.

B. Formato de Arquivos XYZ e FDF

O formato XYZ descreve estruturas atômicas de forma minimalista: a primeira linha contém o número de átomos, a segunda é reservada a comentários e as demais definem coordenadas cartesianas (símbolo, x, y, z). Sua simplicidade garante interoperabilidade, mas é insuficiente para os parâmetros físicos exigidos em simulações quânticas [8].

O FDF é o arquivo principal de entrada do SIESTA. Além das coordenadas, ele define o funcional de correlação e troca — GGA (*Generalized Gradient Approximation*) ou LDA (*Local Density Approximation*), ambas aproximações matemáticas para o termo de troca-correlação da DFT, diferindo na forma como incorporam o gradiente da densidade eletrônica — os limiares de convergência do SCF (*Self-Consistent Field*, ciclo iterativo que resolve as equações de Kohn-Sham até a densidade eletrônica convergir) e os blocos de célula unitária [9]. O *parser* interno do SIESTA aceita *tags* em qualquer ordem e assume padrões para parâmetros omitidos, exigindo apenas os blocos essenciais: `NumberOfAtoms`, `NumberOfSpecies` e `AtomicCoordinatesAndAtomicSpecies`.

Para ilustrar a transformação realizada pela GusSiesta Platform, o Trecho 1 apresenta um arquivo XYZ minimalista para a molécula de água, e o Trecho 2 apresenta o trecho correspondente do arquivo FDF gerado a partir dele. Observa-se que o algoritmo precisa: enumerar as espécies químicas únicas (oxigênio e hidrogênio) e atribuir-lhes um índice; reescrever as coordenadas substituindo o símbolo do elemento pelo índice de espécie correspondente; e inserir os blocos de cabeçalho e de parâmetros de simulação que o arquivo XYZ, por definição, não contém.

Listing 1. Arquivo XYZ de entrada (molécula de água).

```
3
Water molecule
O   0.000000   0.000000   0.000000
H   0.758602   0.000000   0.504284
H   0.758602   0.000000  -0.504284
```

Listing 2. Trecho do arquivo FDF gerado a partir do Trecho 1.

```
NumberOfAtoms      3
NumberOfSpecies    2

%block ChemicalSpeciesLabel
  1   8   O
  2   1   H
%endblock ChemicalSpeciesLabel

AtomicCoordinatesFormat  Ang
%block AtomicCoordinatesAndAtomicSpecies
  0.000000  0.000000  0.000000  1
  0.758602  0.000000  0.504284  2
  0.758602  0.000000 -0.504284  2
%endblock AtomicCoordinatesAndAtomicSpecies
```

Apesar dessa aparente simplicidade, a preparação manual é propensa a erros tipográficos silenciosos — por exemplo, atribuir incorretamente o índice de espécie a uma coordenada — lacuna que a GusSiesta Platform visa preencher por meio da automação descrita na Seção IV-C.

C. Pseudopotenciais

Pseudopotenciais são aproximações matemáticas que substituem os elétrons de caroço e o núcleo por um potencial iônico efetivo (*frozen-core approximation*) [10], explorando o princípio de que as propriedades de ligação química são ditadas pelos elétrons de valência. O SIESTA suporta pseudopotenciais *norm-conserving* (que preservam a norma da função de onda dentro do raio de corte) em arquivos `.psf`, exigindo um arquivo por espécie química presente na simulação. Para garantir reprodutibilidade, a comunidade conta com o repositório PseudoDojo [11] e a biblioteca oficial do SIESTA.

D. Tecnologias Utilizadas

A plataforma baseia-se no *framework* Django, que segue o padrão arquitetural MVT (*Model-View-Template*): o *Model* define a persistência de dados, a *View* concentra a lógica de negócio e o *Template* renderiza a apresentação. A maturidade do Django, sua segurança integrada — incluindo proteção nativa contra CSRF (*Cross-Site Request Forgery*, ataque que induz o usuário autenticado a executar ações não intencionais) — e seu ecossistema Python o tornam adequado para aplicações científicas [12]. A interface usa Bootstrap 5 para responsividade *Mobile-First*. A visualização 3D é feita pela biblioteca 3Dmol.js, que renderiza estruturas moleculares via WebGL diretamente no

navegador, sem *plugins* [13]. O ciclo de vida da aplicação é encapsulado com Docker, que resolve incompatibilidades de ambiente e garante portabilidade e reprodutibilidade no *deploy* [14].

III. Trabalhos Relacionados

Esta seção discute soluções existentes que se relacionam, total ou parcialmente, com o problema enfrentado pela GusSiesta Platform: a ausência de uma ferramenta web que integre conversão de arquivos, visualização molecular e gerenciamento de pseudopotenciais para o SIESTA. As ferramentas analisadas foram selecionadas por pertencerem a uma das três frentes que, combinadas, compõem o fluxo de pré-processamento de uma simulação DFT: visualização e preparação de estruturas moleculares, automação de *workflows* de simulação, e plataformas web de ciência dos materiais. A comparação entre elas (Tabela I) considera quatro critérios diretamente relacionados aos objetivos deste trabalho: disponibilidade como aplicação **Web** (sem exigir instalação local), suporte nativo ao **SIESTA** (geração de arquivos FDF ou integração direta com o simulador), presença de **visualização 3D** interativa, e contribuição para uma **baixa barreira** de entrada a usuários iniciantes.

Na frente de visualização e preparação de estruturas, o VESTA oferece alta qualidade gráfica e suporte a simetrias cristalinas, mas exige instalação local e tem curva de aprendizado elevada [15]; por isso, na Tabela I, recebe suporte apenas *parcial* ao SIESTA (não gera FDF) e baixa contribuição à redução da barreira de entrada. O Avogadro é editor molecular de código aberto com suporte a Gaussian e GAMESS, porém sem suporte nativo ao FDF do SIESTA [16]; ainda assim, por ser instalável localmente com interface gráfica simples, é classificado como de baixa barreira. O JSmol opera no navegador via HTML5, mas, por se tratar de uma biblioteca de renderização e não de uma aplicação completa, não gerencia arquivos nem gera parâmetros de simulação [17].

Na automação de *workflows*, o ASE provê interface unificada para múltiplos calculadores incluindo o SIESTA [18], porém exige linha de comando, elevando a barreira de entrada para estudantes iniciantes. O AiiDA oferece rastreabilidade completa via grafos de proveniência com *plugin* para o SIESTA [19], mas requer a configuração de serviços auxiliares como PostgreSQL e RabbitMQ, tornando sua implantação impraticável em contextos pedagógicos introdutórios.

Entre plataformas web, o Materials Cloud foca na disseminação de resultados e reprodução de cálculos já publicados, não em um fluxo guiado de preparação de novos arquivos [20]; o ioChem-BD concentra-se no pós-processamento e na curadoria de dados de saída [21]; e o nanoHUB oferece ferramentas de simulação isoladas entre si, sem um ambiente integrado ao ciclo de vida de um projeto SIESTA [22].

Tabela I
Comparativo entre ferramentas relacionadas e a GusSiesta Platform

Ferramenta	Web	SIESTA	Viz. 3D	Baixa Barreira
VESTA	Não	Parcial	Sim	Não
Avogadro	Não	Não	Sim	Sim
JSmol	Sim	Não	Sim	Parcial
ASE	Não	Sim	Não	Não
AiiDA	Não	Sim	Não	Não
Materials Cloud	Sim	Parcial	Sim	Não
ioChem-BD	Sim	Não	Não	Parcial
nanoHUB	Sim	Não	Parcial	Parcial
GusSiesta Platform	**Sim**	**Sim**	**Sim**	**Sim**

Como sintetiza a Tabela I, observa-se uma lacuna entre soluções de alta complexidade técnica (AiiDA, ASE) e visualizadores de propósito geral (VESTA, Avogadro, JSmol): nenhuma das ferramentas analisadas reúne, simultaneamente, disponibilidade web sem instalação local, suporte nativo ao SIESTA e visualização 3D integrada. É exatamente essa combinação que a GusSiesta Platform propõe preencher, ao reunir em uma única interface acessível via navegador as etapas de pré-processamento necessárias para o início de uma simulação no SIESTA.

IV. Metodologia e Desenvolvimento do Sistema

O desenvolvimento seguiu os princípios da metodologia ágil XP [5], organizado em ciclos curtos de implementação e validação. As práticas adotadas foram: *small releases* com entregas incrementais ao orientador; *simple design* priorizando a menor estrutura que satisfizesse os requisitos; *refactoring* contínuo para manter a coesão dos módulos; e *testing*, com escrita de testes unitários ao longo do desenvolvimento (detalhados na Seção IV-G). O repositório público no GitHub registra o histórico de evolução do projeto [6].

A. Requisitos do Sistema

Os requisitos foram levantados a partir das necessidades do LASIMON e organizados em funcionais (RF) e não funcionais (RNF), sintetizados nas Tabelas II e III, respectivamente.

B. Arquitetura Geral da Plataforma

A GusSiesta Platform adota o padrão MVT do Django (Seção II). O projeto é organizado em três aplicações Django independentes mais um diretório de dados. O módulo `converter/` concentra a lógica de conversão de XYZ para FDF, a integração com o visualizador 3D e o empacotamento dos pseudopotenciais. O módulo `user/` gerencia autenticação (cadastro, *login*, *logout*). O módulo `dashboard/`, acessível a administradores, exibe um panorama das rotas do sistema para monitoramento. Os arquivos `.psf` residem no diretório `pseudos/`, lidos pelo módulo `converter/` no momento do *download*. A exposição das funcionalidades por meio de uma API REST

Tabela II
REQUISITOS FUNCIONAIS DA GUSSIESTA PLATFORM

ID	Descrição
RF01	O sistema deve permitir o *upload* de arquivos XYZ pelo usuário autenticado.
RF02	O sistema deve converter arquivos XYZ em arquivos FDF compatíveis com o SIESTA.
RF03	O sistema deve exibir a visualização 3D interativa da molécula contida no arquivo XYZ antes da conversão.
RF04	O sistema deve permitir a configuração dos parâmetros de simulação PAO, MD, SCF e XC pelo usuário.
RF05	O sistema deve identificar os elementos químicos presentes no arquivo XYZ e disponibilizar os pseudopotenciais (`.psf`) correspondentes para *download*.
RF06	O sistema deve empacotar o arquivo FDF gerado e os pseudopotenciais em um arquivo `.zip` para *download*.
RF07	O sistema deve oferecer cadastro, autenticação e encerramento de sessão de usuários.
RF08	O sistema deve manter o histórico de conversões realizadas por cada usuário, permitindo consulta e *download* posterior.
RF09	O sistema deve permitir salvar, carregar e excluir configurações de parâmetros de simulação.

Tabela III
REQUISITOS NÃO FUNCIONAIS DA GUSSIESTA PLATFORM

ID	Descrição
RNF01	A aplicação deve ser containerizada com Docker e orquestrada via *docker-compose*, garantindo reprodutibilidade do ambiente.
RNF02	A interface deve ser responsiva, compatível com navegadores modernos em *desktops* e dispositivos móveis, utilizando Bootstrap 5.
RNF03	O sistema deve utilizar SQLite em ambiente de desenvolvimento e PostgreSQL em produção, sem alteração de código.
RNF04	A aplicação deve ser servida em produção por Gunicorn (servidor WSGI, do inglês *Web Server Gateway Interface*, responsável por encaminhar requisições HTTP à aplicação Django), com os arquivos estáticos entregues via WhiteNoise, dentro de contêineres Docker.
RNF05	O código-fonte deve ser mantido publicamente em repositório Git, com histórico de *commits* rastreável.
RNF06	O sistema deve implementar proteção contra acesso não autenticado às rotas de conversão e *download*.
RNF07	O tempo de resposta para a geração do arquivo FDF deve ser inferior a cinco segundos para arquivos XYZ de até 500 átomos.

(*Application Programming Interface* que segue o estilo arquitetural *Representational State Transfer*) está prevista como trabalho futuro (Seção V-D).

O diagrama de casos de uso da Figura 1 sintetiza as interações entre o usuário autenticado, o sistema externo via API (funcionalidade futura, indicada para contraste com as funcionalidades já implementadas) e as funcionalidades disponíveis na plataforma.

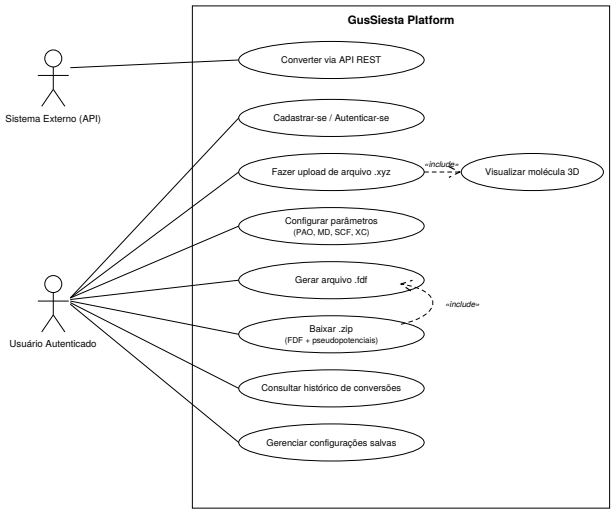

Figure 1. Diagrama de casos de uso da GusSiesta Platform. Fonte: Elaborado pelo Autor.

C. Módulo de Conversão XYZ para FDF

O módulo `converter/` constitui o núcleo funcional da plataforma. A *view* `ConvertView` recebe a requisição HTTP do tipo POST, valida os dados via `SIESTAParametersForm` e aciona a função `convert_xyz_to_fdf`, que implementa o algoritmo de conversão. O Algoritmo 1 descreve, em pseudocódigo, as quatro etapas executadas: leitura do arquivo XYZ; identificação das espécies químicas únicas; montagem dos blocos de parâmetros a partir do formulário; e concatenação final na ordem exigida pelo *parser* do SIESTA [9]. Esse algoritmo é precisamente o que transforma o Trecho 1 no Trecho 2, apresentados na Seção II-B.

A arquitetura de módulos da plataforma é ilustrada na Figura 3: o usuário interage com o *frontend* (Bootstrap 5 e 3Dmol.js), cujas requisições chegam ao servidor Gunicorn; internamente, o Django organiza as funcionalidades nos três módulos descritos na Seção IV-B; os dados de sessão e histórico são persistidos em PostgreSQL (produção), e os arquivos de pseudopotencial são lidos diretamente do sistema de arquivos. A Figura 2 detalha, em um diagrama de sequência, a ordem de troca de mensagens entre esses componentes durante uma conversão completa, incluindo o registro opcional do histórico quando o usuário está autenticado.

D. Parâmetros de Simulação Configuráveis

A plataforma expõe quatro grupos de parâmetros via `SIESTAParametersForm`, correspondentes aos conceitos apresentados na Seção II. O grupo PAO define o tipo de base (`PAO.BasisType`: *split* ou *nodes*) e o seu tamanho (`PAO.BasisSize`: de SZ,

Require: arquivo XYZ, nome do sistema, parâmetros de simulação (PAO, MD, SCF, XC)
Ensure: conteúdo do arquivo FDF
1: $atomos \leftarrow$ LERLINHAS(XYZ a partir da linha 3)
2: $numAtomos \leftarrow$ primeira linha do XYZ
3: $especies \leftarrow$ lista vazia
4: **for all** $atomo$ em $atomos$ **do**
5: **if** símbolo de $atomo \notin especies$ **then**
6: adicionar símbolo de $atomo$ a $especies$
7: **end if**
8: **end for**
9: $rotulos \leftarrow$ ENUMERARESPECIES($especies$) ▷ bloco ChemicalSpeciesLabel
10: $coords \leftarrow$ INDEXARCOORDENADAS($atomos$, $rotulos$)
11: $blocoParam \leftarrow$ MONTARPARAMETROS(PAO, MD, SCF, XC) ▷ a partir do formulário
12: $fdf \leftarrow$ CONCATENAR(cabeçalho, $rotulos$, $coords$, $blocoParam$)
13: **return** fdf

Algoritmo 1: Conversão de XYZ para FDF (função `convert_xyz_to_fdf`)

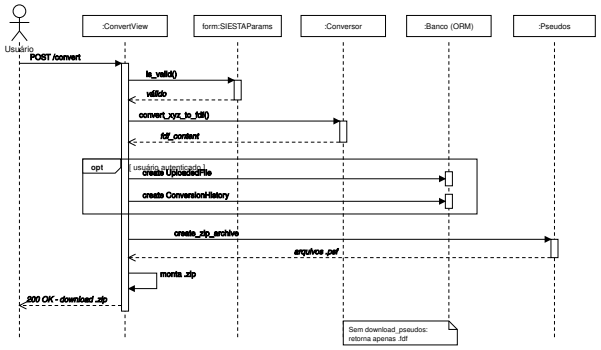

Figure 2. Diagrama de sequência do fluxo de conversão e download (método `ConvertView.post`). Fonte: Elaborado pelo Autor.

Single Zeta, a DZP, *Double Zeta Polarized*); o padrão adotado é DZP, reconhecido na literatura como o melhor compromisso entre custo computacional e precisão para sistemas moleculares de médio porte [3]. O grupo MD controla a otimização geométrica (`MD.TypeOfRun`, `MD.NumCGsteps`, `MD.MaxForceTol`). O grupo SCF define o número máximo de iterações do ciclo autoconsistente (Seção II-B) e o critério de convergência de energia (`SCF.DM.Tolerance`). O grupo XC seleciona o funcional de troca e correlação — LDA ou GGA/PBE (padrão) —, este último amplamente adotado em simulações de nanoestruturas de carbono e silício [1].

E. Visualizador 3D e Download de Pseudopotenciais

A visualização 3D é implementada via 3Dmol.js integrada ao *template* `upload.html` [13]. Após o *upload*, o conteúdo do arquivo XYZ é passado ao objeto `viewer` por meio de uma variável JavaScript embutida no *template*, renderizando a estrutura molecular imediatamente via WebGL, sem necessidade de *plugins*. O usuário pode rotacionar, transladar e aplicar *zoom*, além de alternar entre os estilos de representação *stick*, *sphere* e *line*. Esse *feedback* visual imediato permite verificar a integridade da estrutura antes da conversão, conforme ilustrado na Figura 7 (Seção V-A).

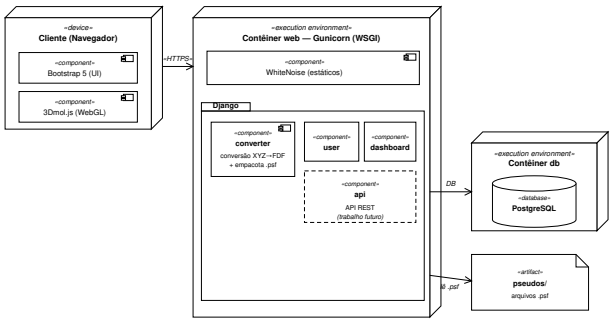

Figure 3. Diagrama de implantação (*deployment*) da GusSiesta Platform: contêineres Docker representados como *nodes*, módulos Django como componentes internos e a API REST (tracejada) como trabalho futuro. Fonte: Elaborado pelo Autor.

O empacotamento dos pseudopotenciais, implementado na função `create_zip_archive`, opera em três etapas: extração dos símbolos químicos únicos do arquivo XYZ; localização dos arquivos `.psf` correspondentes no diretório `pseudos/`, derivados do PseudoDojo [11] e da biblioteca oficial do SIESTA, no formato *norm-conserving* [10]; e compressão do arquivo FDF e dos `.psf` em um único arquivo `.zip`, com a estrutura de diretórios esperada pelo SIESTA para execução direta.

F. Autenticação e Containerização

O módulo `user/` usa o subsistema nativo `django.contrib.auth` [12] para cadastro, *login* e *logout*. As rotas de conversão são protegidas pelo decorador `@login_required` (RNF06), e as senhas são armazenadas com hash PBKDF2-SHA256 — algoritmo de derivação de chave (*Password-Based Key Derivation Function 2*) que aplica a função de resumo criptográfico SHA-256 repetidamente, dificultando ataques de força bruta sobre as senhas armazenadas — mecanismo padrão do Django.

A containerização usa Docker orquestrado por `docker-compose` (RNF01) [14], com dois serviços: `web` (Django via Gunicorn, com WhiteNoise servindo os arquivos estáticos) e `db` (PostgreSQL em produção). O script `entrypoint.sh` executa automaticamente as migrações do banco antes de iniciar o Gunicorn, garantindo a sincronização do esquema de dados ao subir o contêiner.

```
$ pytest -tb=short -v
======== test session results ========
collected 88 items

converter/tests.py::test_read_xyz_atoms
PASSED
converter/tests.py::test_read_xyz_coordinates
PASSED
converter/tests.py::test_xyz_symbols_unchanged
PASSED
...
(88 testes executados em 56.7s)

======== 88 passed in 56.71s ========
```

Figure 4. Saída resumida da execução dos testes com `pytest`. Fonte: Elaborado pelo Autor.

G. Testes

A estratégia de testes combinou testes unitários automatizados com `pytest-django` e testes manuais funcionais, em conformidade com a prática de *testing* da metodologia XP (Seção IV). A suíte totaliza 88 cenários distribuídos em `converter/tests.py` (46 testes), `user/test_views.py` (35 testes) e `dashboard/test_views.py` (7 testes), cobrindo a leitura e validação de arquivos XYZ, a geração dos blocos obrigatórios do FDF, o empacotamento do arquivo `.zip` e o ciclo completo de autenticação. A Figura 4 apresenta a saída resumida da execução da suíte, que totaliza 56,7 segundos com 100% de aprovação.

Os testes manuais, conduzidos pelo autor e pelo orientador, cobriram o fluxo completo via navegador: cadastro, *upload* e renderização 3D de moléculas de referência (incluindo a Heparina), configuração dos parâmetros PAO/MD/SCF/XC, geração do arquivo `.zip`, histórico de conversões e comportamento em modo containerizado.

H. Histórico de Conversões e Configurações Salvas

O módulo `converter/` implementa dois modelos de dados que estendem a plataforma além da conversão pontual. O modelo `ConversionHistory` persiste cada conversão realizada, armazenando o conteúdo do arquivo FDF gerado, os parâmetros de simulação em formato JSON (*JavaScript Object Notation*, um formato textual leve para troca de dados estruturados), a contagem de *downloads* e a data da operação. A *view* `history_view`, acessível apenas ao usuário autenticado, exibe os registros paginados e permite o *download* de conversões passadas sem repetir o processo. O modelo `SavedConfiguration` permite salvar, carregar e excluir conjuntos de parâmetros (PAO, MD, SCF, XC) com nome identificador, reduzindo o retrabalho em simulações recorrentes. Esses dois modelos e seus relacionamentos com o modelo de usuário do Django são apresentados no diagrama de classes da Figura 5.

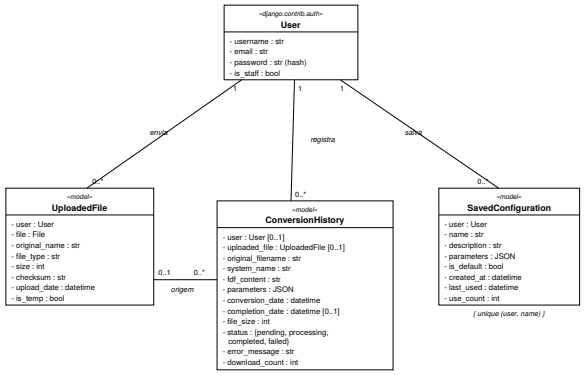

Figure 5. Diagrama de classes dos modelos de dados da GusSiesta Platform. Fonte: Elaborado pelo Autor.

Tabela IV
MÉTRICAS DE QUALIDADE DA GUSSIESTA PLATFORM

Métrica	Resultado
Testes automatizados	88
Testes aprovados	88
Taxa de sucesso	100%
Tempo médio de conversão (XYZ→FDF) para 105 átomos	< 1 s *
Tempo médio de geração do pacote ZIP (FDF + PSF)	< 1 s *

* Valores estimados em ambiente de desenvolvimento local.

V. RESULTADOS E DISCUSSÃO

Esta seção apresenta os resultados obtidos com a GusSiesta Platform, organizados em quatro frentes: demonstração do fluxo de uso da plataforma (Seção V-A), validação estrutural do arquivo FDF gerado (Seção V-B), avaliação preliminar da plataforma (Seção V-C) e discussão das limitações identificadas (Seção V-D).

A Tabela IV resume os indicadores quantitativos obtidos a partir da suíte de testes (Seção IV-G) e da validação manual (Seção V-B). Os 88 testes automatizados foram aprovados integralmente, e os tempos de resposta permaneceram muito abaixo do limite de 5 segundos estabelecido no requisito RNF07.

A. Demonstração da Plataforma

A plataforma foi implantada e validada em ambiente containerizado, acessível via navegador sem necessidade de instalação local. O fluxo principal de uso, ilustrado nas Figuras 6 a 9, compreende cinco etapas. Na primeira etapa (Figura 6), o usuário realiza o cadastro e a autenticação na plataforma, requisito necessário para que o histórico de conversões (Seção IV-H) seja registrado. Na segunda etapa (Figura 7), imediatamente após o *upload* do arquivo XYZ, a plataforma exibe a visualização 3D interativa da molécula via 3Dmol.js, permitindo a inspeção da estrutura antes da conversão, conforme descrito na

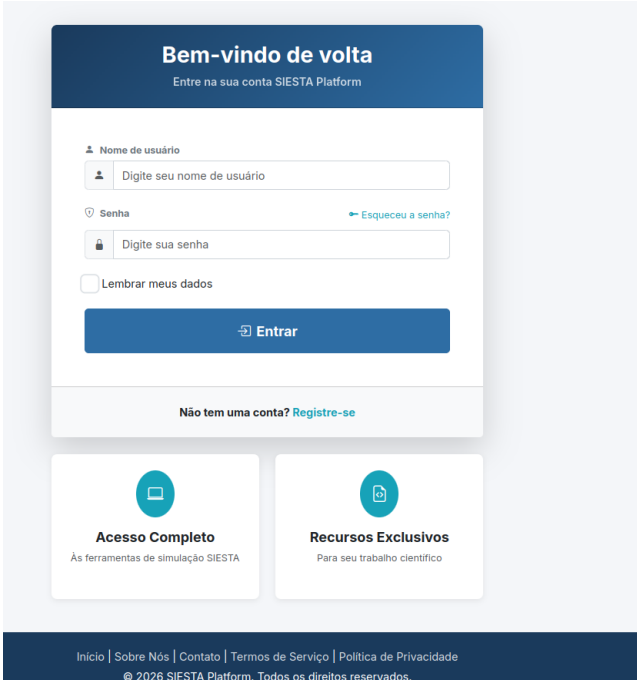

Figure 6. Tela de autenticação da GusSiesta Platform. Fonte: Elaborado pelo Autor.

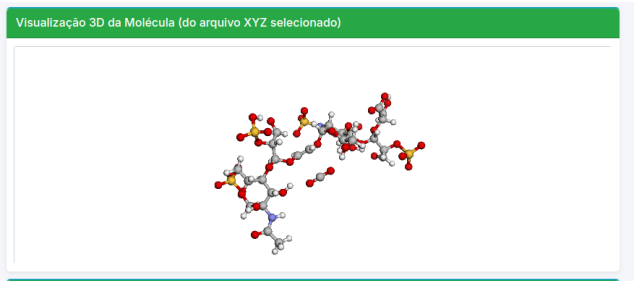

Figure 7. Visualização 3D interativa da molécula de Heparina (arquivo `Heparin.xyz`) renderizada via 3Dmol.js. Fonte: Elaborado pelo Autor.

Seção IV-E. Na terceira etapa (Figura 8), o usuário configura os parâmetros de simulação por meio do formulário `SIESTAParametersForm` (Seção IV-D). Na quarta etapa, a plataforma executa o Algoritmo 1 para gerar o arquivo FDF e empacota os pseudopotenciais correspondentes aos elementos químicos identificados. Na quinta e última etapa (Figura 9), o arquivo `.zip` resultante, contendo o FDF e os `.psf` correspondentes, é disponibilizado para *download* imediato.

B. Validação do Arquivo FDF Gerado

A validação estrutural do arquivo FDF gerado pela plataforma foi realizada utilizando a molécula de Heparina como caso de referência: o arquivo `Heparin.xyz` contém 105 átomos distribuídos entre as espécies químicas carbono (C), hidrogênio (H), nitrogênio (N), oxigênio (O) e enxofre

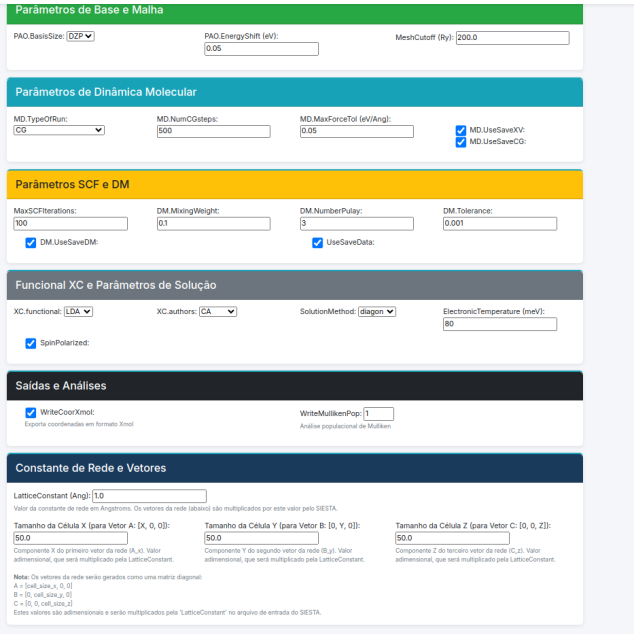

Figure 8. Formulário de configuração dos parâmetros de simulação PAO, MD, SCF e XC da GusSiesta Platform. Fonte: Elaborado pelo Autor.

(S). A escolha dessa molécula deveu-se à sua relevância para os trabalhos do LASIMON e à diversidade de espécies químicas que ela contém em uma única estrutura, permitindo testar simultaneamente a enumeração de múltiplas espécies pelo Algoritmo 1. A inspeção manual do arquivo FDF gerado confirmou a presença e a corretude de todos os blocos obrigatórios [9]: `NumberOfAtoms`, `NumberOfSpecies`, o bloco `ChemicalSpeciesLabel` com os índices corretos para cada espécie química, o bloco `AtomicCoordinatesAndAtomicSpecies` com as coordenadas e os índices de espécie devidamente mapeados — seguindo exatamente o padrão de transformação ilustrado nos Trechos 1 e 2 —, e os parâmetros de simulação configurados pelo usuário.

A correspondência entre `NumberOfAtoms` e o total de linhas de coordenadas do `Heparin.xyz`, bem como entre `NumberOfSpecies` e as espécies únicas extraídas pelo algoritmo, foi verificada e confirmada. Esses resultados indicam que o algoritmo de conversão produz, ao menos para o caso testado, arquivos estruturalmente válidos e prontos para uso direto no SIESTA. Cabe ressaltar, contudo, que essa validação foi realizada com uma única molécula; a Seção V-D discute essa limitação e a necessidade de validação com um conjunto mais amplo e diverso de estruturas moleculares.

C. Avaliação Preliminar da Plataforma

Uma avaliação preliminar da GusSiesta Platform foi conduzida pela própria equipe de desenvolvimento — o autor e o orientador, ambos com experiência prévia no uso

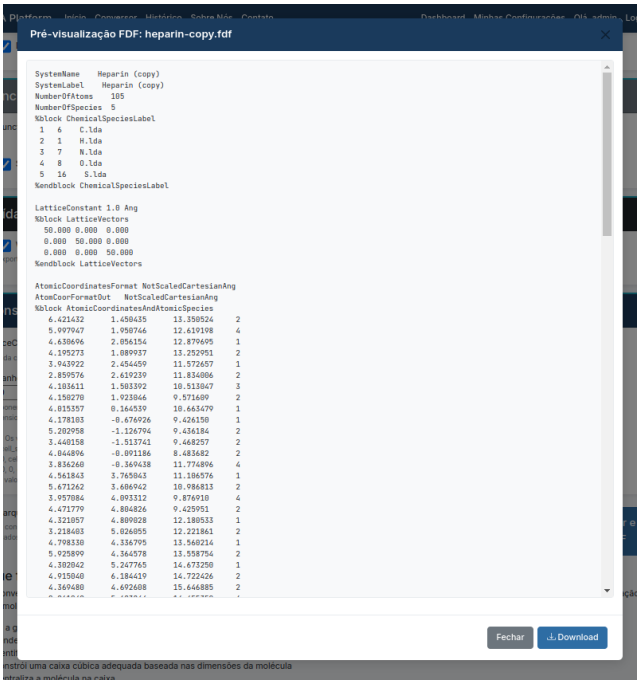

Figure 9. Página de resultado com o arquivo `.zip` disponível para *download*, contendo o arquivo FDF gerado e os pseudopotenciais (`.psf`) correspondentes aos elementos da molécula carregada. Fonte: Elaborado pelo Autor.

```
NumberOfAtoms   105
NumberOfSpecies 5

%block ChemicalSpeciesLabel
1   6   C
2   1   H
3   7   N
4   8   O
5   16   S
%endblock ChemicalSpeciesLabel

AtomicCoordinatesFormat   Ang
%block AtomicCoordinatesAndAtomicSpecies
1.234   0.567   0.890   1   # exemplo
-0.234   1.890   2.345   2
...   (demais 103 átomos)
%endblock AtomicCoordinatesAndAtomicSpecies
```

Figure 10. Trecho do arquivo FDF gerado pela plataforma para a molécula de Heparina (105 átomos), seguindo a mesma estrutura de blocos do exemplo didático da Seção II-B. Fonte: Elaborado pelo Autor.

do SIESTA — por meio da execução sistemática do fluxo completo (autenticação, submissão de arquivo, configuração de parâmetros e *download*) com coleta informal de *feedback* ao longo das sessões. É importante destacar que essa avaliação tem caráter exploratório e não constitui um teste de usabilidade formal: não houve participação de usuários externos à equipe de desenvolvimento, nem aplicação de instrumentos padronizados de medição.

Dentro dessas limitações, o fluxo completo foi executado sem dificuldades técnicas, o que constitui um indício — não uma comprovação — de que a interface pode contribuir para reduzir a barreira de entrada ao SIESTA. Os principais pontos qualitativos observados foram: (i) a visualização 3D foi percebida como funcionalidade de alto valor para verificação imediata da estrutura; (ii) a organização dos parâmetros em grupos nomeados (PAO, MD, SCF, XC) foi considerada clara, mas pressupõe alguma familiaridade prévia do usuário com a nomenclatura do SIESTA; e (iii) a entrega do arquivo `.zip` completo foi percebida como redução do tempo de preparação de uma simulação, em comparação ao processo manual descrito na Seção II-B, embora essa comparação não tenha sido medida quantitativamente.

Para sustentar de forma robusta a afirmação de que a plataforma reduz a barreira de entrada, são necessários, como trabalho futuro: testes com usuários externos sem envolvimento no desenvolvimento; comparação cronometrada entre a preparação manual e a preparação assistida pela plataforma para uma mesma molécula; e aplicação de instrumentos padronizados de usabilidade, como a *System Usability Scale* (SUS).

D. Limitações e Trabalhos Futuros

A GusSiesta Platform apresenta limitações que motivam desenvolvimentos futuros, organizadas aqui da mais para a menos crítica do ponto de vista metodológico.

Em primeiro lugar, a validação estrutural do FDF gerado (Seção V-B) e a avaliação preliminar de uso (Seção V-C) foram realizadas com uma única molécula e por uma equipe de apenas duas pessoas diretamente envolvidas no desenvolvimento. Essa limitação reduz a robustez das conclusões sobre a corretude geral do algoritmo de conversão frente a moléculas com geometrias, tamanhos e espécies químicas distintas, bem como sobre a real redução da barreira de entrada para um público mais amplo de usuários. A ampliação do conjunto de moléculas de teste e a condução de avaliações com usuários externos são, portanto, extensões prioritárias deste trabalho.

Em segundo lugar, os pseudopotenciais são referenciados de forma fixa como `.lda.psf`, independentemente do funcional XC selecionado pelo usuário no formulário (Seção IV-D): quando o usuário escolhe GGA/PBE, os arquivos entregues ainda são do tipo LDA, podendo introduzir inconsistências em cálculos que exijam alta precisão. A integração dinâmica com o PseudoDojo [11] e a inclusão de variantes GGA/PBE são extensões necessárias.

Em terceiro lugar, o processamento da conversão ocorre de forma síncrona na *thread* principal do servidor Gunicorn (Seção IV-F). Para arquivos XYZ com centenas de átomos ou para uso simultâneo por múltiplos usuários, essa abordagem pode introduzir latência perceptível; a adoção de filas de tarefas assíncronas com Celery (sistema de

filas distribuídas para execução de tarefas em segundo plano) associado ao Redis (banco de dados em memória frequentemente usado como intermediário de mensagens) ou com Django-RQ (integração mais leve entre Django e a biblioteca *Redis Queue*) é a evolução natural para garantir escalabilidade.

Em quarto lugar, a plataforma não expõe API REST para acesso programático ao conversor, limitando a integração com *scripts* de laboratório e ferramentas externas de automação. A implementação de *endpoints* RESTful é um dos trabalhos futuros prioritários, aproveitando a lógica já concentrada no módulo `converter/` (Seção IV-B).

Por fim, o conversor reconhece aproximadamente 13 elementos químicos; espécies fora dessa lista podem gerar número atômico inválido no arquivo FDF sem erro explícito ao usuário — a Heparina não é afetada por essa limitação, mas a extensão da tabela periódica suportada e a inclusão de validação explícita são melhorias necessárias. Adicionalmente, conversões realizadas por usuários anônimos geram registros de histórico com o campo `user` nulo, que permanecem inacessíveis pela interface (que exige autenticação para consulta); e o campo `email_verified`, presente no modelo de perfil de usuário, não é atualmente definido por nenhum fluxo da aplicação. A extensão natural mais ambiciosa da plataforma seria a análise dos próprios resultados de saída do SIESTA — incluindo submissão do cálculo a um servidor de computação, monitoramento do *job* e visualização de estrutura de bandas eletrônicas e densidade de estados (DOS, do inglês *Density of States*) —, transformando a GusSiesta Platform em um ambiente completo para o ciclo de vida de uma simulação DFT.

VI. Conclusão

Este trabalho apresentou o desenvolvimento da GusSiesta Platform, uma aplicação web de código aberto que automatiza a conversão de arquivos XYZ para FDF, etapa frequentemente considerada um gargalo na preparação de simulações com o SIESTA. A plataforma integra, em uma única interface acessível via navegador, o *upload* de coordenadas, a visualização 3D interativa, a configuração guiada dos principais parâmetros DFT e o empacotamento automático dos pseudopotenciais necessários — combinação que, conforme discutido na Seção III, não foi identificada em nenhuma das ferramentas correlatas analisadas.

Os objetivos propostos foram cumpridos: o algoritmo de conversão foi implementado e validado estruturalmente com a molécula de Heparina (105 átomos); a biblioteca 3Dmol.js foi integrada com sucesso, fornecendo *feedback* visual imediato; o gerenciamento de pseudopotenciais foi implementado, reunindo no *download* os arquivos necessários para iniciar uma simulação; e a containerização com Docker garante portabilidade e reprodutibilidade do ambiente de execução. A plataforma conta ainda com 88 testes automatizados, com 100% de aprovação, e com funcionalidades adicionais não previstas nos objetivos iniciais — o histórico de conversões e o repositório de configurações salvas — que aumentam a produtividade do usuário recorrente.

Este trabalho também reconhece, de forma explícita, os limites do que foi comprovado: a validação do algoritmo de conversão restringe-se a uma única molécula, e a percepção de redução da barreira de entrada apoia-se em uma avaliação preliminar conduzida pela própria equipe de desenvolvimento, não em testes com usuários externos ou instrumentos formais de usabilidade. Tais lacunas, somadas à ausência de uma API REST e ao tratamento ainda fixo dos pseudopotenciais em LDA, compõem a agenda de trabalhos futuros detalhada na Seção V-D.

Dessa forma, a GusSiesta Platform contribui, sobretudo, como prova de conceito tecnicamente validada para a democratização do acesso ao SIESTA em contextos de ensino de graduação e iniciação científica em simulação de materiais, estando disponível como código aberto e pronta para implantação experimental em instituições de ensino por meio de contêineres Docker, mediante a condução das validações adicionais aqui apontadas.

Referências

[1] N. Marzari, A. Ferretti, and C. Wolverton, "Electronic-structure methods for materials design," *Nature Materials*, vol. 20, pp. 736–749, 2021.

[2] R. O. Jones, "Density functional theory: Its origins, rise to prominence, and future," *Reviews of Modern Physics*, vol. 87, no. 3, pp. 897–923, 2015.

[3] J. M. Soler, E. Artacho, J. D. Gale, A. García, J. Junquera, P. Ordejón, and D. Sánchez-Portal, "The SIESTA method for *ab initio* order-N materials simulation," *Journal of Physics: Condensed Matter*, vol. 14, no. 11, pp. 2745–2779, 2002.

[4] LASIMON – Laboratório de Simulação Molecular, UFN, "Lasimon – laboratório de simulação molecular," https://lasimon.vercel.app/, acessado em: abril de 2025.

[5] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd ed. Addison-Wesley, 2004.

[6] G. G. Pereira, "SIESTA Platform – repositório," https://github.com/GustavoGarciaPereira/SIESTA_Platform, acessado em: março de 2026.

[7] W. Kohn and L. J. Sham, "Self-consistent equations including exchange and correlation effects," *Physical Review*, vol. 140, no. 4A, pp. A1133–A1138, 1965.

[8] F. Jensen, *Introduction to Computational Chemistry*, 3rd ed. Wiley, 2017.

[9] SIESTA Project, "Siesta manual," https://siesta-project.org/siesta/, acessado em: março de 2025.

[10] N. Troullier and J. L. Martins, “Efficient pseudopotentials for plane-wave calculations,” *Physical Review B*, vol. 43, no. 3, pp. 1993–2006, 1991.

[11] M. J. Van Setten *et al.*, “The pseudodojo: Training and grading a 85 element optimized norm-conserving pseudopotential table,” *Computer Physics Communications*, vol. 226, pp. 39–54, 2018.

[12] Django Software Foundation, “Django documentation,” https://docs.djangoproject.com/, acessado em: março de 2025.

[13] N. Rego and D. Koes, “3dmol.js: molecular visualization with webgl,” *Bioinformatics*, vol. 31, no. 8, pp. 1322–1324, 2015.

[14] D. Merkel, “Docker: lightweight linux containers for consistent development and deployment,” *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[15] K. Momma and F. Izumi, “Vesta 3 for three-dimensional visualization of crystal, volumetric and morphology data,” *Journal of Applied Crystallography*, vol. 44, no. 6, pp. 1272–1276, 2011.

[16] M. D. Hanwell *et al.*, “Avogadro: an advanced semantic chemical editor, visualization, and analysis platform,” *Journal of Cheminformatics*, vol. 4, no. 1, p. 17, 2012.

[17] S. Chen, “Jmol/jsmol interactive scripting and digital molecular models,” *Journal of Chemical Education*, vol. 91, no. 1, pp. 147–148, 2014.

[18] A. H. Larsen *et al.*, “The atomic simulation environment—a python library for working with atoms,” *Journal of Physics: Condensed Matter*, vol. 29, no. 27, p. 273002, 2017.

[19] S. P. Huber *et al.*, “Aiida 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance,” *Scientific Data*, vol. 7, no. 1, p. 300, 2020.

[20] L. Talirz *et al.*, “Materials cloud, a platform for open computational science,” *Scientific Data*, vol. 7, no. 1, p. 299, 2020.

[21] M. Álvarez Moreno *et al.*, “iochem-bd: a device-agnostic molecular data management system,” *Journal of Chemical Information and Modeling*, vol. 55, no. 1, pp. 95–103, 2015.

[22] K. Madhavan *et al.*, “nanohub.org: Advancing education and research in nanotechnology,” *Computing in Science & Engineering*, vol. 15, no. 2, pp. 8–19, 2013.