TÍTULO

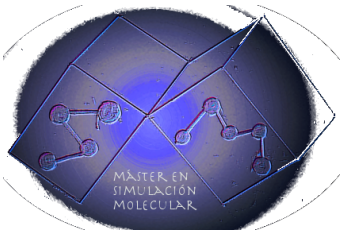
DOCUMENTATION & INTEGRATION OF AUTOMATED WORKFLOW IN ATOMISTIC SOFTWARE FOR CHARGE ANALYSIS WITH SIESTA

AUTOR

Ramón María Bergua López

Esta edición electrónica ha sido realizada en 2023

Tutores	Dr. D. Felipe Jiménez Blas; Dra. Dª. Mónica García-Mota
Instituciones	Universidad Internacional de Andalucía; Universidad de Huelva
Curso	*Máster en Simulación molecular (2021-2022)*
©	Ramón María Bergua López
©	De esta edición: Universidad Internacional de Andalucía
Fecha documento	2022



Documentation & Integration of Automated Workflow in Atomistic Software for Charge Analysis with SIESTA

Ramón María Bergua López

Trabajo entregado para la obtención del grado de Master en Simulación Molecular

Julio / 2022

Directores:

Dr. Felipe Jiménez Blas

Dra. Mónica García-Mota

RESUMEN

En esta memoria se describe el trabajo llevado a cabo durante una estancia de 3 meses en SIMUNE Atomistics, una empresa que provee de herramientas de simulación a la comunidad científica, académica e industrial de todo el mundo, y está desarrollando un software de simulación en Python llamado ASAP[1] (Atomistic Simulation Advanced Platform). ASAP contiene, entre otros paquetes, el uso del código abierto de simulación SIESTA[2,3,4] (Spanish Initiative for Electronic Simulations with Thousands of Atoms), para facilitar al usuario el trabajo de manera mucho más rápida y flexible a través de una interfaz gráfica evitando así el uso de la terminal. También entre otros servicios, SIMUNE ofrece tutoriales específicos de SIESTA así como apoyo profesional para aquellos usuarios del código que quieran resolver problemas de computación específicos.

El objetivo de mi estancia en SIMUNE consistió en documentar en forma de tutorial la manera de llevar a cabo el análisis de cargas mediante el uso de SIESTA que se incluirá en los tutoriales que ofrecen como parte de sus servicios a clientes. Una vez completado, tuve que implementar en ASAP una función que permitiera automatizar el análisis de cargas a través del software.

Los pasos a seguir para completar mi objetivo satisfactoriamente fueron los siguientes. Tuve que familiarizarme con el uso del código SIESTA en la terminal, completando para ello los tutoriales de uso ofrecidos por SIMUNE y realicé un trabajo de feedback aportando mi experiencia de usuario para la mejora del mismo. Más adelante, estudié los distintos métodos de análisis de carga disponibles en SIESTA que calculan la población electrónica de una sustancia. Estos son los métodos de Mulliken, Hirshfeld y Voronoi y aprendí en qué consisten y por qué motivo es más recomendable el uso de un método u otro. Busqué en el manual de SIESTA la manera de utilizarlos y propuse una serie de ejemplos sencillos.

Al finalizar la redacción del tutorial me encargué de la implementación de dicha funcionalidad en el software ASAP. Para ello tuve que escribir un código de programación orientado a objetos con el lenguaje Python. Comenzando por el diseño del widget que permite al usuario seleccionar entre los tres distintos métodos de análisis, continuando con el tratamiento de datos del archivo de salida que emite

SIESTA para extraer el resultado del análisis y finalizando con la exposición de los mismos en una tabla en la que el usuario pueda visualizarlos. Para este trabajo fue vital el uso de GitLab, un portal de trabajo que permite a los equipos de desarrollo trabajar en el código, a través de ramas que se modifican de manera independiente y que luego se incluyen en el código principal tras su revisión.

Como parte de mi tarea de desarrollador del software, fue necesario incluir también unos tests que verifiquen que la aplicación funcione como se espera, mediante modelos sencillos introducidos manualmente en el código, y cuyos resultados conocidos se comparaban con los resultados emitidos por la aplicación. Además, dado que los resultados de un cálculo atomístico varían en función de la multiplicidad de espín electrónico seleccionada (molécula no polarizada, spin colineal, o no colineal), tuvimos que tenerlo en cuenta a la hora del tratamiento de datos y exposición de resultados. En última instancia, aplicamos cambios cosméticos que dotaban a la tabla de una estética más elegante y legible para el usuario y presentamos el resultado en una demo a todo el equipo de la empresa.

ABSTRACT

This report describes the work carried out during a 3-month internship at SIMUNE Atomistics, a company that provides simulation tools to the scientific, academic and industrial community around the world. Simune is developing a simulation software in Python called ASAP (Atomistic Simulation Advanced Platform). ASAP contains, among other packages, the use of the open source code SIESTA (Spanish Initiative for Electronic Simulations with Thousands of Atoms), to facilitate the user's work in a much faster and more flexible way through an interface graph thus avoiding the use of the terminal. In addition, among other services, SIMUNE offers specific SIESTA tutorials as well as professional support for those code users who want to solve specific computer problems.

The goal of this internship was to document as a tutorial about how to perform charge analysis using SIESTA, which will be included in the tutorials that SIMUNE offers as part of their services. Once completed, I had to implement a function in ASAP that automatize charge analysis inside the software.

The steps to follow were the following. I get comfortable with the use of SIESTA code in terminal, by fulfilling user tutorials offered by SIMUNE and doing a feedback work to contribute with my user experience to improve those tutorials. Then, I studied the different charge analysis methods available in SIESTA that calculate electronic population for a given substance. These methods are the Mulliken, Hirshfeld and Voronoi methods and I learned about their principles and also how to perform them using SIESTA wondering which method is more suitable for any situation.

After writing the tutorial I was in charge of the implementation of said functionality in the ASAP software. For this task I had to develop an object-oriented programming code with the Python language. Starting with the design of the widget that allows the user to select between the three different analysis methods, continuing with the data processing of the output file issued by SIESTA, extract the result of the analysis and a presentation in a tablewhere the user can see them. Vital to this work was the use of GitLab, a web portal that allows development teams to work on code, through branches that are modified independently of the master code. These branches can be merge into master code after review.

As part of my task as a software developer, it was also necessary to include some tests that verify that the application works as expected, through simple models introduced manually in the code, and whose known results were compared with the

results issued by the application. In addition, given that the results of an atomistic calculation vary depending on the electron spin multiplicity selected (unpolarized molecule, collinear spin, or non-collinear), we had to take this into account when processing the data and presenting the results. Ultimately, we applied cosmetic changes that gave the table a more elegant and user-readable aesthetic and presented the result in a demo to the entire company team.

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN

1.1. Estado del arte

La computación científica lleva años desarrollando métodos numéricos[1,2,3] que tratan de arrojar luz sobre el comportamiento a pequeña escala de numerosos procesos naturales a los que no pueden alcanzar las ciencias experimentales. Esta disciplina ha evolucionado en paralelo junto con los avances informáticos que permiten cálculos complejos cada vez más rápidos. Debido a la gran importancia que han cobrado estos métodos computacionales en la actualidad, existen a día de hoy una gran variedad de paquetes de simulación[4,5,6] que permiten la investigación científica con distintos niveles de detalle, desde las simulaciones de grano grueso en la meso-escala, hasta la computación cuántica pasando por simulaciones a nivel molecular, clásicas o atomísticas. Estas últimas, requieren de la resolución de las ecuaciones electrónicas para describir el movimiento y las interacciones entre los núcleos atómicos. Uno de los paquetes más utilizados para esta técnica es la herramienta de código abierto SIESTA[7], que mediante la Teoría del Funcional de la Densidad (DFT) permite simular sistemas a escala atómica con una gran flexibilidad computacional, es decir, desde exploraciones rápidas y aproximadas, hasta simulaciones costosas que llegan a proporcionar un gran nivel de detalle.

Dado que el uso de este paquete, y de muchos otros, requieren conocimientos previos en el uso de la terminal, existen empresas que dedican sus esfuerzos en implementar en un software de simulación, este tipo de herramientas, de manera que faciliten al usuario el uso del código, sin la necesidad de usar la terminal y con la agilidad y versatilidad de un programa de ventanas que esté disponible en cualquier sistema operativo. Este es el caso de Simune Atomistics, una *joint venture* del CIC Nanogune en Donostia/San Sebastián que está formado por un equipo internacional y multidisciplinar que, entre otros servicios, ofrece la licencia de uso del programa que desarrolla con el nombre de Atomistic Simulations Advanced Platform (ASAP)[8]. Este software permite al usuario una manera de trabajar mucho más rápida sencilla y eficaz.

1.2. Atomistic Simulation Advanced Platform (ASAP)

ASAP se compone de un sistema de gestión de proyectos (Figura 1), ya sea mediante la creación de uno nuevo con acceso a una base de datos de moléculas, sólidos, superficies, o clústers de uso común, o bien mediante la posibilidad de importar proyectos propios de sistemas específicos. Incluye herramientas de visualización estructural y gráfica (Figura 1 a), todo ello en un sistema intuitivo de interfaz mediante ventanas donde el usuario puede seleccionar de manera sencilla las características de la simulación que quiere llevar a cabo y correrla. Permite realizar cálculos de dinámica molecular, optimización de estructura, obtención de ecuación de estado (Figura 1 c), etc. ya sea mediante el uso del código SIESTA 4.0 y SIESTA 4.1, ASE (Atomic Simulation Environment), o EMT (Electronic Medium Theory) o TRANSIESTA (Electronic Transport SIESTA) (FIGURA 1 (e)).

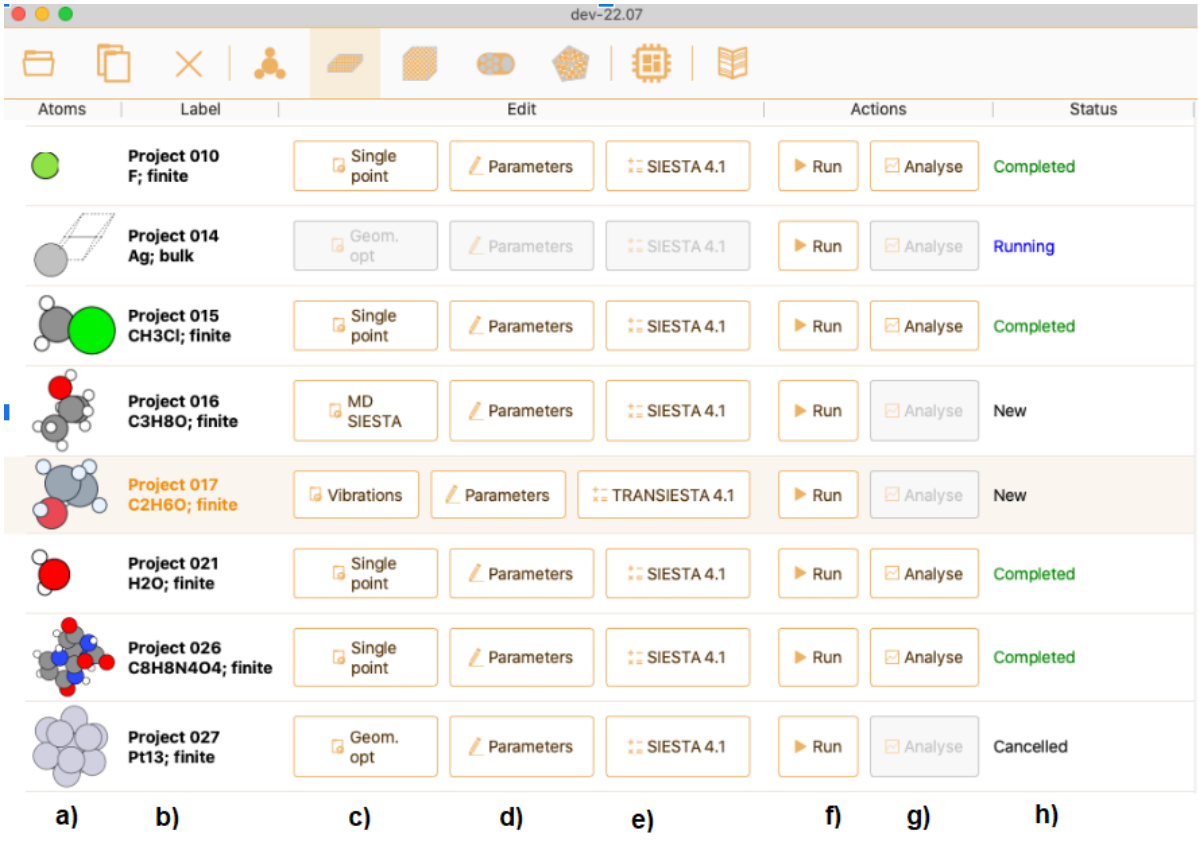


Figura 1. Ventana principal del Software ASAP, cada una de las secciones horizontales corresponde a un proyecto, podemos observar de izquierda a derecha: (a) una representación gráfica de la sustancia, (b) la descripción de cada proyecto de simulación, (c) botón de selección del tipo de cálculo, (d) los parámetros de simulación, (e) el paquete de simulación escogido, (f) las opciones para correr la simulación, (g) la sección de análisis y (h) el estado de cada proyecto.

En la sección de parámetros podemos incluir o deseleccionar de nuestro cálculo algunas opciones como el momento magnético, la densidad local de orbitales (LDOS) o la estructura de bandas electrónicas, en esta sección será donde implementaremos la opción que permitirá al usuario incluir en los resultados de su simulación el análisis de cargas de la sustancia. También podemos personalizar las opciones de simulación en el ámbito informático, (Figura 1, f) como correr en local o en un servidor, en serie o en paralelo y el número de cores que asignamos a la misma. También contamos con el botón de análisis que se activa una vez ha terminado la simulación(Figura 1, g) y donde podremos visualizar los resultados, en forma de gráficas, tablas, etc. Por último la etiqueta que nos indica el estado de la simulación que puede ser nueva, si aun no se ha lanzado; completada, si ya ha terminado; corriendo, si aún se encuentra realizando los cálculos; cancelada, si el usuario impidió la finalización de manera manual; o fallida, si por algún motivo la simulación no terminó debido a algún problema.

1.3. Importancia del análisis de carga

La estructura electrónica de los materiales controla sus propiedades, y la distribución de carga en una molécula, sólido o sustancia en general, es un aspecto fundamental de dicha estructura[9]. Cuando describimos una sustancia y colocamos sobre cada átomo una carga parcial como resultado de los aspectos cualitativos como afinidad electrónica o electronegatividad sobre un átomo dado estamos simplificando esta caracterización de la distribución de carga para hacernos una idea visual de la deslocalización de la nube electrónica, y la naturaleza de los enlaces que conforman esa sustancia[10,11,12,13,14,15,16,17,18,19]. Con esta información podremos entender fenómenos físicos observados en dicha sustancia tales como las propiedades magnéticas, la transferencia electrónica, polarizabilidad, tan útiles en química orgánica para elegir el lugar de ataque de una sustitución nucleófila, en bioquímica con el plegado de proteínas mediante fuerzas electrostáticas o en química inorgánica al explicar las interacciones de los compuestos de coordinación con ligandos neutros.

Es por este motivo que las cargas atómicas parciales se usan a menudo en modelización y cuyo desarrollo ha dado lugar numerosas aproximaciones y métodos

para calcular y determinar dichas cargas. Uno de los pioneros en esta materia fue Mulliken que propuso un método de caracterización de estas cargas a partir de las funciones de onda en 1955 [20] , sin embargo a día de hoy se sabe que los valores obtenidos por este método no son necesariamente realistas, que su resultado depende mucho del tamaño de la base elegida y que los valores pueden llegar a desviarse mucho de los esperados cuando tratamos sistemas especialmente grandes. La siguiente cuestión a plantear entonces es, qué método usar para obtener un resultado de cargas parciales que sea fidedigno. Existen numerosos modelos que permiten determinar la carga atómica neta a partir de la función de onda y pueden dividirse en tres grandes grupos. Métodos de análisis poblacional, el cual incluye el análisis de Mulliken, el de Löwdin[21] , y el de cargas naturales, por otro lado tenemos los métodos de partición de densidad electrónica, como el método de Bader, (también conocido como *atoms-in-molecules*)[22,23,24] el de Hirshfeld[25], métodos DDEC (Density-Derived Electrostatic and Chemical), CM5 (Charge Model 5) o el VDD (Voronoi Deformation Density) propuesto por Baerends et al. que también trataremos con detalle más adelante, por último los métodos de ajuste electrostático, como el método de Merz-Kollman[26].

El grupo de Wang et al. en 2014 estudió complejos metálicos polares en fase gas. utilizando 10 modelos de carga distintos[27]. También Martin y Zipse realizaron un estudio comparativo de cinco métodos propuestos para estudiar la molécula de agua[28], comparando estos con resultados experimentales. Del mismo modo Choudhuri y Truhlar cuando aplicaron esta metodología para comparar distintas metodologías en sólidos[29] o Sigfridsson y Ryde[30] que hicieron lo propio en moléculas.

Por este motivo, dada la necesidad de una gran parte de la comunidad científica que requiere el cómputo de la población de carga atómica de las distintas sustancias que se encuentren bajo estudio, se decidió implementar está característica en el software ASAP.

1.4. Tipos de análisis de carga implementados en SIESTA

De todos los métodos mencionados anteriormente, vamos a prestar especial atención a los tres métodos incluidos en el Manual de SIESTA en la versión 4.1 que emiten sus resultados directamente en el archivo de output. Estos son los métodos de Mulliken, Hirshfeld y Voronoi. Aunque SIESTA permite también calcular el análisis de cargas de Bader, pero este requiere un posprocesado más costoso por lo que la dirección de este proyecto decidió comenzar con estos otros. A continuación vamos a describir brevemente el tipo de análisis de carga con los que se trabaja.

1.4.1 Análisis de población electrónica de Mulliken

El análisis de carga de Mulliken está basado en la Combinacion Lineal de Orbitales Atómicos (LCAO) y por tanto depende de la función de onda del sistema.[31, 32, 33, 34] Fue descrito por R.S. Mulliken en 1955 y la idea principal de este método se basa en que el orbital molecular (MO) esté definido como una combinación lineal de los orbitales atómicos (AO). Cada AO tiene asignado un número de electrones y tras la combinación de orbitales, obtenemos una matriz de densidad cuya diagonal indica la población de los MO.

El resultado tras la integración sobre todos los orbitales es lo que se conoce como población atómica neta, mientras que la población de cada uno de los orbitales que lo componen se denomina población de solapamiento (*overlap population*).

El análisis de población de Mulliken ha sido históricamente el más importante de todos los métodos basados en la función de onda, pese a que sus resultados son fuertemente dependientes de la base molecular elegida. A pesar de esa gran deficiencia, a día de hoy se sigue utilizando debido a su facilidad de cálculo.

1.4.2 Análisis de población electrónica de Hirshfeld

El análisis de cargas por el método de Hirshfeld fue desarrollado por el autor que le dio su nombre en 1977[25]. Las cargas de Hirshfeld se producen con la deformación de la densidad de carga, donde dicha deformación se define como la diferencia que existe entre la densidad de carga final de la molécula y la de los átomos libres.

Para una descripción cuantitativa de la distribución de carga, se divide el sistema en regiones atómicas bien definidas. Para escoger adecuadamente esa división, se decide compartir la densidad de carga en los puntos donde la distancia entre átomos es proporcional a la distancia correspondiente a la densidad de carga si los átomos estuvieran libres. De esta manera, se produce la distribución de carga de los átomos enlazados en las regiones definidas donde cada una de ellas se parece mucho a la densidad molecular en dicha región.

La carga atómica neta, se obtiene entonces al integrar las densidades de deformación atómicas (densidad de carga en el enlace frente a la densidad de carga en el átomo libre) así como el momento multipolar, que en conjunto resultan en la reorganización de la carga de la molécula. De esta manera se puede calcular el potencial electrostático externo y con él la energía de interacción intermolecular o intramolecular.

1.4.3 Análisis de población electrónica de Voronoi

El método de análisis de carga de Voronoi, desarrollado por Baerends et al. en 2003 y denominado VDD[35] (Voronoi Deformation Density method) surge a partir de la idea de distribución espacial en celdas no solapadas, modeladas según el diagrama planteado por Georgy Voronoi en 1908[36, 37] y que ha dado lugar a numerosas aplicaciones en ciencias[38, 39, 40].

1.5. Objetivos

La finalidad de estas prácticas en Simune Atomistics es acercarme al entorno laboral en el ámbito científico llevando a cabo por un lado, la redacción y documentación en forma de tutorial del análisis de cargas de una molécula ejemplo (acetileno), mediante el uso de código abierto SIESTA a través de los métodos de Mulliken, Hirshfeld y Voronoi. Proveer al usuario de las instrucciones detalladas para la construcción del fichero input, y la localización y extracción rápida de los resultados en el fichero de output.

Por otro lado, mi labor más importante como parte del equipo de desarrollo de software de Simune durante estas prácticas, consiste en la implementación de una nueva función en ASAP que permita al usuario incluir el análisis de cargas en su simulación. La función, al activarse, debe mostrar una pantalla de selección donde el usuario pueda seleccionar entre los tres tipos de análisis mencionados. Al finalizar la simulación, el usuario debe poder visualizar los resultados en una tabla en forma de ventana emergente que funcione de manera óptima.

CAPÍTULO 2. MATERIALES Y MÉTODOS

2.1 Redactar Tutorial para SIESTA

Para enseñar a alguien a hacer algo, has de saber hacerlo tú primero(x), por ese motivo, el primer paso para la redacción del tutorial fue dirigirme el manual de usuario de SIESTA, y encontrar las instrucciones que activan los distintos análisis de cargas para que se incluyen en el fichero input y obtener resultados. A continuación realicé distintas pruebas con moléculas ejemplo y para ello fue imprescindible el uso de la terminal y del lenguaje bash, pues el código SIESTA está diseñado para correr simulaciones a través de ella.

Una vez comprendido el procedimiento para realizar un análisis de cargas y encontrado en el fichero de salida los resultados, preparé un directorio con los archivos necesarios y redacté un tutorial paso a paso con todos los comandos necesarios para que un usuario que desconoce el entorno de la terminal pueda llevar a cabo un análisis de cargas a través del código SIESTA.

Como ejemplo para el tutorial opté por la molécula de acetileno, por lo que fue necesario adquirir los pseudopotenciales de los átomos de Carbono e Hidrógeno, requeridos por SIESTA y escogimos aquellos que proveen en la base de datos del ICMAB[41].

Por último, dado el carácter científico, y por tanto internacional, del producto, este tutorial fue redactado en inglés mediante el uso de LaTex, con la inestimable ayuda del equipo.

2.2 Implementar Análisis de Carga en ASAP

Uso de Gitlab

El desarrollo de ASAP se realiza en GitLab. Una de las principales plataformas de desarrollo de Software cuya principal característica es que permite modificar simultáneamente, de manera eficiente y segura, el código fuente de un programa mediante un sistema de ramas.

Para interactuar con el repositorio de GitLab utilizamos una serie de comandos específicos como: *git pull*, para clonar la rama en tu máquina y trabajar de manera local; *git add*, para añadir los cambios realizados en tu versión local; *git push*, para subir la versión modificada de vuelta a la nube. Además estos comandos pueden ir acompañados de opciones para añadir etiquetas a los cambios, o tratar con varios archivos a la vez.

Trabajo en equipo

Durante esta parte de las prácticas, fue imprescindible comprender el método de trabajo de un equipo de desarrollo de software. Es muy importante destacar que la clave de este método se basa en el trabajo en equipo, por ese motivo todas las mañanas nos reuníamos en un *daily meeting*. De esta manera, poner en común los avances y las trabas con las que nos habíamos encontrado cada uno durante la jornada anterior y así todo el equipo podía aportar con distintos puntos de vista e ideas, sobre el trabajo que desarrollabas. Dada la importancia de este factor, a partir de este momento la redacción de esta memoria la continuaré en plural.

Método de trabajo

Para el sistema de modificaciones en el código seguimos una metodología concreta. Se parte, en primer lugar, de la versión original llamada *master*, será el tronco principal del código y se encuentra en un repositorio de GitLab. A partir de ahí seguimos un proceso esquemático de aplicación de cambios que sigue una estructura jerárquica de tareas.

Se propone una sección (*epic)* que se divide en pequeñas etiquetas (*issues*) que se irán resolviendo de una en una siguiendo el orden establecido. Para aplicar las modificaciones de cada *issue* se utiliza un sistema de ramas (*branches*) donde al comenzar un *issue* se abre una nueva rama que modificaremos en paralelo al código. Estas ramas al completarse, se someten a revisión por parte de cualquier otro miembro del equipo, el cuál chequea, no solo la funcionalidad de la nueva rama, sino la limpieza del código y su correcta documentación, de esta manera será más sencillo el mantenimiento del código en un futuro si una tercera persona tiene que modificarlo.

Una vez revisada, se fusiona la rama en el código principal y se cierra el *issue*. Para realizar cambios en el siguiente *issue*, se sigue el mismo procedimiento. Se abre otra rama desde master, se aplican los cambios, se revisa, y se fusiona de nuevo con la rama principal o *master*.

Este sistema de trabajo permite a los desarrolladores trabajar en varias ramas simultáneamente, aplicando cambios en distintas partes del código. Además, gracias a la revisión de un segundo miembro evitamos trasladar errores al código principal. Se puede observar en la Figura 2 una visión esquemática del procedimiento descrito.

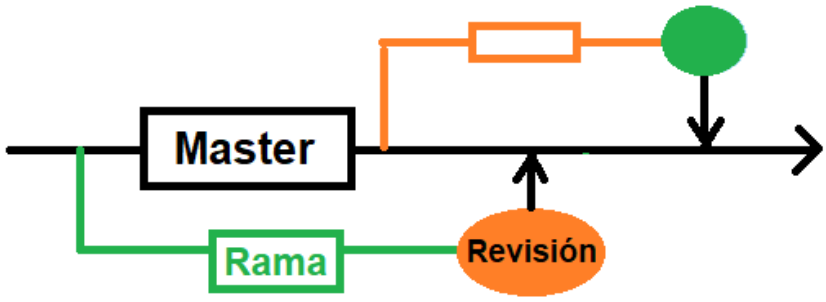


Figura 2. Descripción esquemática del sistema de modificación de código usando Gitlab. En negro se muestra la rama principal o *master*, la línea verde representa una rama de modificaciones que revisa el usuario naranja y valida la fusión en la rama principal. Simultáneamente, el desarrollador naranja trabaja en su rama y cuando la termina, el desarrollador verde verifica que puede unirla a la rama *master*.

En el Anexo I he incluido los distintos *issues* que me asignaron en GitLab y que conformaban el *epic* Charge Analysis.

Python, QtPy, Pytest

El código del Software ASAP está escrito y desarrollado en el lenguaje de programación Python, el cual es un lenguaje ampliamente utilizado por la comunidad de desarrolladores por su versatilidad, facilidad de sintaxis y sobre todo por las numerosas librerías disponibles en la red que hacen de Python uno de los principales lenguajes de programación a día de hoy en el mundo.

Este lenguaje permite programar las instrucciones internas(*back end*) que efectúan el tratamiento de los datos obtenidos en la simulación, así como el diseño de los widgets, botones e imágenes (*front end*) con las que el usuario interactúa cuando utiliza ASAP. Para diseñar estas ventanas hemos utilizado la plataforma QtDesigner que utiliza la librería PyQt para configurar la apariencia que tendrá la ventana.

Por último, para evitar que las continuas modificaciones aplicadas en el software, provoquen errores de flujo en el mismo, es necesario incluir unos tests que verifiquen que lo que ya funcionaba no se vea afectado por los cambios aplicados en el código. Para esta tarea recurrimos a la librería Pytest, la cuál incluye funcionalidades muy útiles a la hora de recorrer el código y testear que la aplicación se comporta con normalidad cada vez que aplicamos una modificación. Además diseñamos los tests que van a recorren la parte nueva del código, de esta manera las futuras modificaciones que se efectúen sobre el código, se verán obligadas a cumplir con los tests ya escritos. Aunque el diseño de estos tests no afecta al comportamiento del programa y *a priori* puede parecer una tarea extra, este procedimiento es muy útil tanto a la hora del mantenimiento y conservación del código, como en la detección de errores.

Editor de texto

Por último en cuanto a herramientas utilizadas cabe destacar el uso del editor de texto plano Virtual Studio Code (VSCode), dadas las facilidades que ofrece para la gestión de archivos, y la gran cantidad de opciones que se le pueden instalar a modo de *plug-ins* ya sean formateadores de texto, funciones de autocompletado, interpretadores de lenguaje, etc. Además VSCode ofrece una función *debugger* que permite correr el script paso a paso para una mayor limpieza del código y facilidad en la corrección de errores.

CAPÍTULO 3. RESULTADOS Y DISCUSIÓN

3.1 Plan de Implementación

Como hemos comentado en la sección anterior el sistema de modificaciones sigue una estructura esquemática, donde un *epic* engloba los diferentes *issues* relacionados con el mismo. En mi caso el *epic* se denominaba *Charge Analysis* por razones evidentes y se dividía en cuatro *issues* principales: el diseño del *front end* de selección. que se refiere al diseño de la ventana o *widget* donde el usuario selecciona el análisis de carga que quiere realizar de entre todos los modelos disponibles; también el diseño del *backend* de selección, es la parte del código en la que utilizamos las preferencias seleccionadas por el usuario y las introducimos en el flujo de trabajo de ASAP y que incluya en el fichero input de SIESTA los comandos que activan el análisis de cargas durante la simulación; el diseño del *backend* de resultados, en el que se extraen los datos del fichero de salida que provee SIESTA al finalizar la simulación, este se postprocesa, depura y filtra para almacenar los resultados obtenidos en la memoria del programa; y el diseño del *front end* de resultados, en forma de ventana emergente diseñamos un widget que contiene una tabla con los resultados obtenidos tras la simulación.

3.1.1 Diseño de Widget selección y backend de selección

Cuando el usuario abre el programa y abre un proyecto para efectuar una simulación puede escoger entre varias opciones, parámetros, etc. Para ello debe navegar por la aplicación seleccionando y deseleccionando las distintas características para adecuar la simulación a sus necesidades.

Entre estas opciones disponibles se encuentra el análisis de cargas, el cual vamos a ubicar en la sección *Parameters*. Al clicar sobre esta sección se despliega una ventana, en cuyo menú aparecen las distintas funciones disponibles y al entrar en el apartado de *Charge Analysis* podremos pedir a través de un sistema de *checkbox*, (cajas de selección), si queremos que la simulación incluya el análisis de Mulliken, de Hirshfeld, de Voronoi o de varios de ellos simultáneamente. (Figura 3) El diseño de esta ventana se realizó mediante el programa QtDesigner que utiliza las librerías PyQt5 y Pyside2.

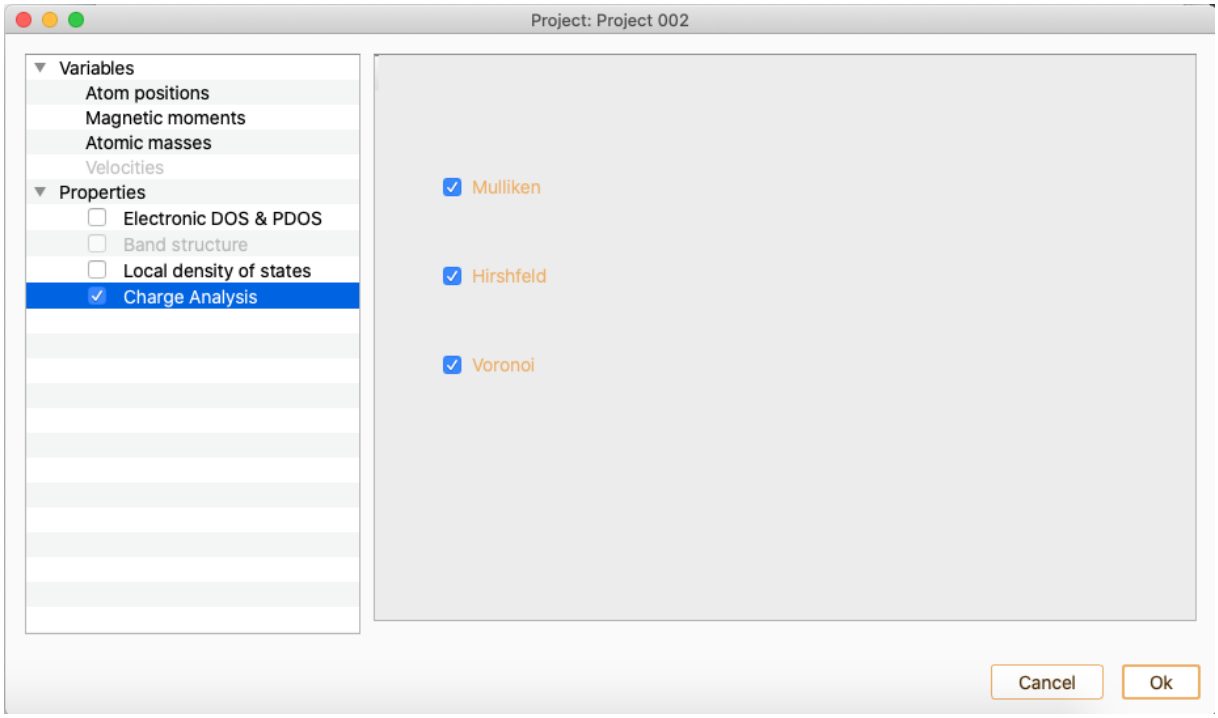


Figura X. Menú de selección de las propiedades de simulación. A la izquierda se puede observar que el análisis de cargas ha sido activado, y por ese motivo en la sección derecha, podemos observar que los tres tipos de análisis, Mulliken, Hirshfeld y Voronoi, se activan por defecto.

Ahora tendremos que diseñar la parte del código en la que recabamos las opciones seleccionadas por el usuario y que se almacenan en la memoria en forma de *booleanos (true* o *false)*. Estas variables se trasladan a la parte del código en la que se configura el fichero de input que utiliza SIESTA para correr la simulación (Figura 4). En nuestro caso, al archivo de input le añadimos el comando asociado que le indicará a SIESTA si durante la simulación tendrá que realizar el análisis de carga o bien de Mulliken, o bien de Hirshfeld, o de Voronoi, o bien de dos de ellos, o de todos, en función del valor de los booleanos elegidos por el usuario.

```
WriteHirshfeldPop   True
WriteMullikenPop    1
WriteVoronoiPop     True
```

Figura 4. Comandos que se añaden sobre el archivo input de SIESTA que activa el estudio del análisis de carga.

Aunque en una primera instancia puede parecer sencillo, fue necesario configurar los valores por defecto de las variables, así como el almacenamiento de las mismas

tras la selección del usuario. Comprobamos que tras abrir y cerrar los distintos menús, se conservan las opciones elegidas por el usuario y no se revierten los cambios a los valores establecidos por defecto.

Una vez que el usuario ha definido claramente su proyecto y decide correr la simulación, solo nos quedará recoger los resultados emitidos por SIESTA y presentarlos al usuario.

3.1.2 Backend de postprocesado de resultados

En esta sección, tendremos que diseñar un pequeño script que permita extraer los resultados y almacenarlos en variables. En primer lugar vamos a leer el fichero en el que SIESTA imprime los resultados de la simulación. Utilizamos una función de la librería Numpy que permite la lectura y almacenaje del contenido de un archivo en forma de cadenas de caracteres (*strings)*.

A partir de aquí identificamos la sección del archivo en la que se muestran los resultados del análisis de cargas de Mulliken (Figura 5, a) y por otro lado el de Hirshfeld y Voronoi (Figura 5, b), pues vienen descritos de manera diferente. Una vez encontrados, aislamos los valores numéricos asociados a la carga de cada átomo. Por último, almacenamos en variables internas del programa los valores obtenidos de cada tipo de análisis para su posterior exposición en una tabla.

```
mulliken: Atomic and Orbital Populations:

Species: C.gga.pbe.1
Atom  Qatom  Qorb
               2s      2s      2py     2pz     2px
               2dxy    2dyz    2dz2    2dxz    2dx2-y
   1  4.575   1.150   0.101   0.975   0.975   0.975
              0.008   0.008   0.000   0.008   0.000

Species: H.gga.pbe.2
Atom  Qatom  Qorb
               1s      1s      1py     1pz     1px
   2  0.856   0.499   0.322   0.012   0.012   0.012
   3  0.856   0.499   0.322   0.012   0.012   0.012
   4  0.856   0.499   0.322   0.012   0.012   0.012
   5  0.856   0.499   0.322   0.012   0.012   0.012

mulliken: Qtot =        8.000
```

```
Hirshfeld Net Atomic Populations:
Atom #    Qatom  Species
     1   -0.152  C.gga.pbe.1
     2    0.038  H.gga.pbe.2
     3    0.038  H.gga.pbe.2
     4    0.038  H.gga.pbe.2
     5    0.038  H.gga.pbe.2

Voronoi Net Atomic Populations:
Atom #    Qatom  Species
     1   -0.097  C.gga.pbe.1
     2    0.024  H.gga.pbe.2
     3    0.024  H.gga.pbe.2
     4    0.024  H.gga.pbe.2
     5    0.024  H.gga.pbe.2
```

Figura 5. Resultados del análisis de cargas de una molécula de metano en el archivo emitido por SIESTA al finalizar la simulación.

Es muy importante a la hora del diseño optimizado de un script tener en cuenta que debe funcionar de forma genérica, es decir, independientemente de que métodos de análisis se hayan escogido, el algoritmo debe ser capaz de encontrar los resultados en el archivo y a su vez que no pierda el tiempo en buscar los resultados de los métodos que no fueron seleccionados.
Una vez encontrados y aislados los valores de carga sobre cada átomo, se asignan a variables en la memoria del programa para a continuación mostrar los resultados en una tabla.

3.1.3 Diseño de Tabla para mostrar resultados

En esta sección hemos de generar una nueva ventana emergente que contenga la tabla en la que se mostrarán los resultados de análisis de carga que hemos aislado en variables anteriormente. Para configurar la apariencia de la ventana hemos recurrido de nuevo a la aplicación QtDesigner.

Para que el usuario pueda visualizar la tabla, tenemos que añadir en la sección de *Analyse* (Figura 1, g) un botón que haga emerger la ventana que contiene a la tabla. (Figura 6)

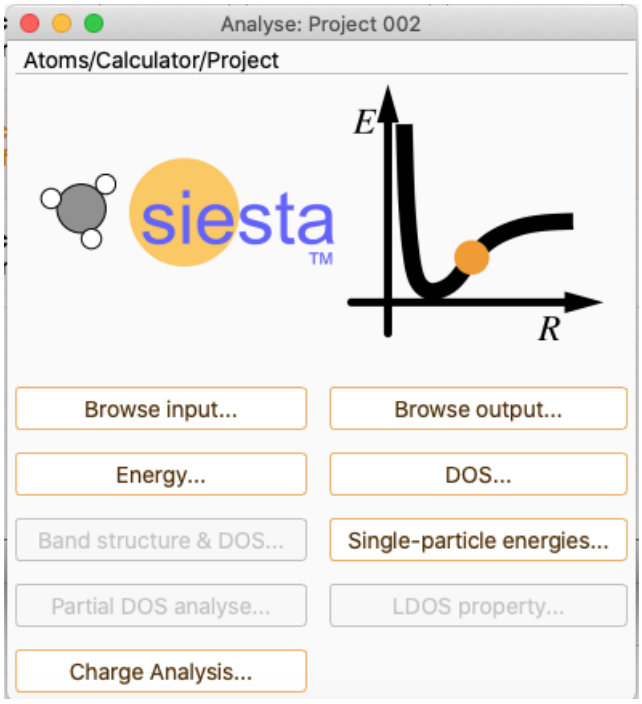


Figura 6. Ventana emergente que permite al usuario visualizar los resultados de la simulación. Observamos abajo a la izquierda el botón implementado Charge Analysis. Cuando el usuario clica sobre ese botón aparece la tabla de resultados en forma de una nueva ventana emergente.

Esta última debe mostrar los resultados por columnas para cada método de análisis, así como incluir las columnas que identifican al índice y el símbolo del átomo al que le corresponde dicha carga. Por tanto, para generar la tabla tenemos que conocer primero el número de elementos que la van a componer, determinando el número de columnas con las que va a contar la tabla en función de los métodos de análisis elegidos, así como de filas, que corresponden al número total de átomos. Rellenamos las celdas de la tabla mediante un bucle que recorre todas las variables que contienen la información obtenida anteriormente.

3.1.4 Diseño de Test que evalúen comportamiento

Como hemos comentado en el apartado de Materiales y métodos, el desarrollo de nuevas funcionalidades en el software de ASAP requiere de la implementación de una serie de test que chequean que el programa va a funcionar con normalidad. Se manera que cada vez que modificamos el programa y

Para ello, se formula un ejemplo cuya respuesta ya se conoce y una vez que el código ha devuelto la respuesta se compara con el valor esperado. Si el código funciona bien y el test está bien desarrollado obtendremos valores idénticos en la respuesta, mientras que si los valores son distintos, significa que o bien el código no está trabajando como esperamos o que el test no se diseñó correctamente.

Esta herramienta es muy útil ya que permite establecer un control de calidad automático sobre el código, ya que cuando añadamos modificaciones, mejoras, o nuevas características al programa que afecten al comportamiento esperado del mismo, los tests evaluarán de nuevo que todo lo desarrollado con anterioridad sigue funcionando como se espera y que las nuevas modificaciones no han interferido con esas otras partes del código.

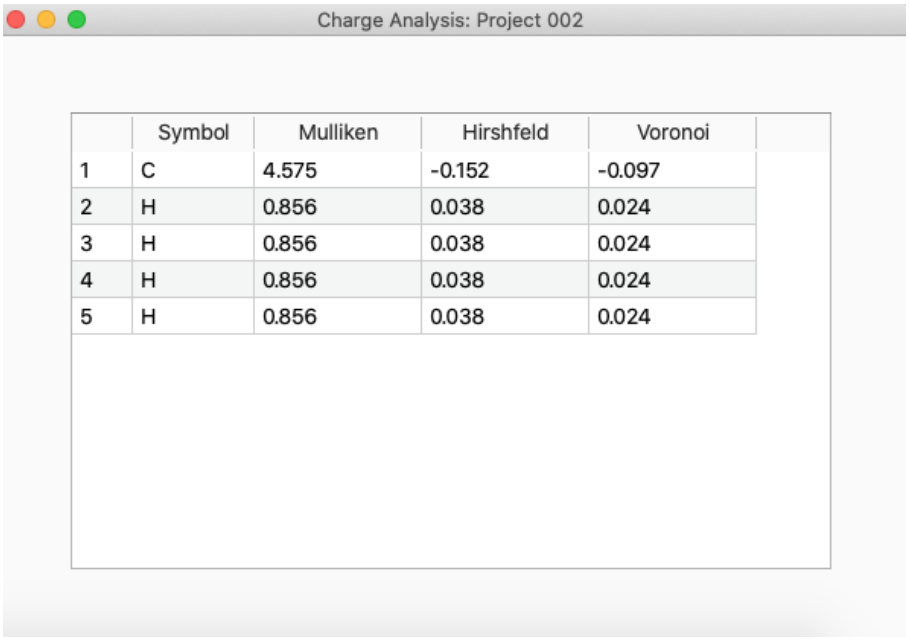
Si al añadir una nueva modificación algún test no pasa, significa que las nuevas instrucciones están afectando a un valor de una variable que ya no es el valor esperado. Cada vez que se sincroniza la versión modificada localmente con la versión almacenada en la nube, todos los tests del software se corren automáticamente para comprobar que las nuevas modificaciones siguen verificando los valores esperados.

3.2 Correcciones

Una vez desarrollada prácticamente en su totalidad la nueva funcionalidad de Análisis de Carga en ASAP, dedicamos la etapa final de los tres meses de mi estancia en prácticas en Simune para tratar de perfeccionar todo lo relativo a la experiencia de usuario. En esta sección trataremos las modificaciones que hicimos a posteriori sobre el código, enfocando dichos cambios a los casos de uso mayoritarios, es decir, orientando esta nueva funcionalidad a la comodidad del usuario.

3.2.1 Normalización de Mulliken

Como ya hemos comentado en la sección anterior, el análisis poblacional de Mulliken se centra en el cálculo del número de electrones que se encuentran sobre uno de los núcleos de la molécula, mientras que el análisis de Hirshfeld y Voronoi, estudia la población relativa, es decir, la cantidad de carga que se encuentra en exceso o defecto sobre la que sería la población natural del átomo en estado neutro. (Figura 7)



	Symbol	Mulliken	Hirshfeld	Voronoi
1	C	4.575	-0.152	-0.097
2	H	0.856	0.038	0.024
3	H	0.856	0.038	0.024
4	H	0.856	0.038	0.024
5	H	0.856	0.038	0.024

Figura 7. Tabla de resultados en la que se muestran los valores de carga calculados para cada átomo en la molécula de metano. En la primera columna se muestra el índice de cada átomo; en la segunda, el símbolo del elemento químico; en la tercera los resultados de Mulliken; y en la cuarta y quinta, los resultados de Hirshfeld y Voronoi respectivamente. Nótese que aún no se han implementado los cambios estéticos que ajustan el tamaño de la ventana al de la tabla (Sección 3.2.4)

Este fenómeno implica que la comparación de resultados sea más sencilla entre los métodos de Hirshfeld y Voronoi, pues representan la población electrónica neta, es decir, la carga 'extra' que presentan los núcleos en comparación con la que tendrían con el estado neutro. Así el defecto de electrones genera una carga positiva y el exceso de electrones se indica con una carga negativa sobre el núcleo.

Para que la comparación con el método de Mulliken sea factible, hemos de sustraer el número de electrones que encontraríamos en la última capa del átomo en estado neutro; esto es, el número atómico Z. Para conocer el número atómico de cada átomo, hemos recurrido al fichero de salida de SIESTA donde se indica para cada especie este valor. Además ya que un exceso de carga se representa con signo negativo, y el exceso con signo positivo, al valor resultante le aplicamos un cambio de signo y de esta manera ya es posible comparar los resultados obtenidos por los tres métodos, de un solo vistazo (Figura 8).

Charge Analysis: Project 006

Atom index	Symbol	Mulliken	Hirshfeld	Voronoi
1	C	-0.575	-0.152	-0.097
2	H	0.144	0.038	0.024
3	H	0.144	0.038	0.024
4	H	0.144	0.038	0.024
5	H	0.144	0.038	0.024

Figura 8. Resultados de análisis de carga de una molécula de metano tras la normalización de Mulliken

3.2.2 Multiplicidad de espín

En primer lugar, cabe destacar que SIESTA es un paquete de simulación a nivel atomístico, es decir, durante la simulación realiza cálculos cuánticos para determinar las fuerzas generadas entre los núcleos. Por ello, encontraremos diferencias en los resultados obtenidos, según la multiplicidad de espín que hayamos fijado.

Podemos restringir la simulación donde los electrones están apareados, es decir cuando la molécula no está polarizada. Este es el caso general y para el cual hemos desarrollado esta funcionalidad. Sin embargo, SIESTA permite simular la molécula desacoplando a los electrones (Figura 9), es decir, con la molécula polarizada atendiendo a dos casos, espín colineal o no colineal, es decir, aunque los electrones estén desapareados podremos encontrarlos con espín paralelo o no.

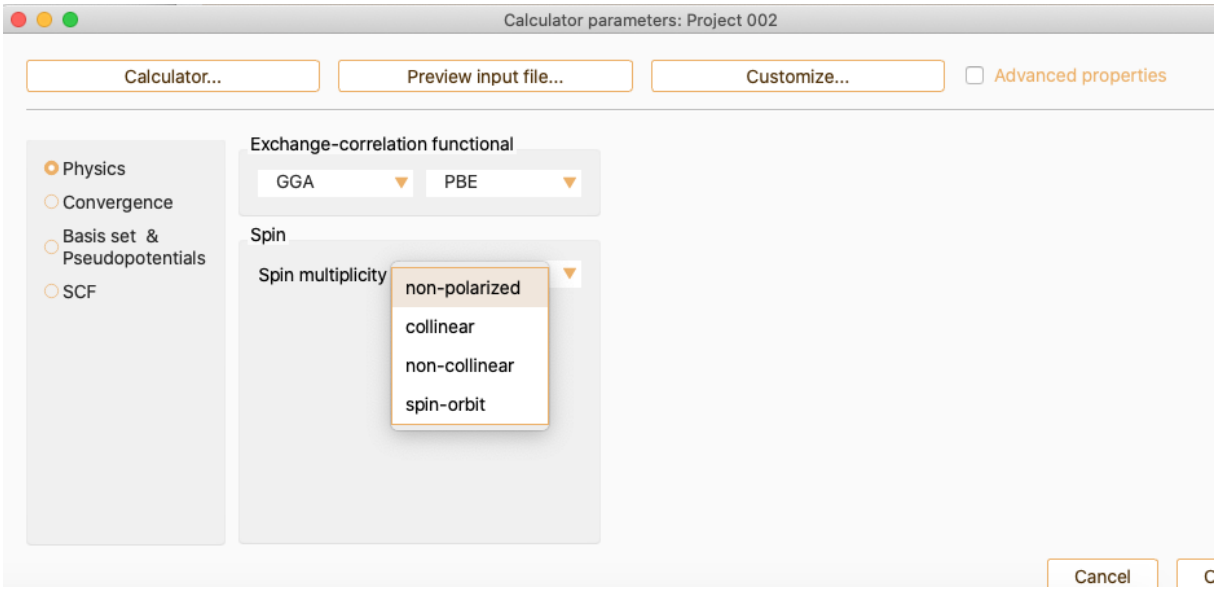


Figura 9. Ventana de selección de las características de simulación. Se puede observar el menú desplegable que permite al usuario elegir la multiplicidad de espín deseada con la que se van a realizar los cálculos.

Esta variación en los resultados hemos de tenerla en cuenta, ya que en el caso del análisis de cargas de Mulliken, que estudia el valor absoluto de la población electrónica sobre cada núcleo, diferencia entre el número de electrones con espín hacia arriba (*spin up*) o espín hacia abajo (*spin down*). Figura 10.

Charge Analysis: Project 007

Atom index	Symbol	Mulliken Up	Mulliken Down	Hirshfeld	Voronoi
1	C	-0.288	-0.288	-0.152	-0.097
2	H	0.072	0.072	0.038	0.024
3	H	0.072	0.072	0.038	0.024
4	H	0.072	0.072	0.038	0.024
5	H	0.072	0.072	0.038	0.024

FIGURA 10. Resultados del análisis de carga de una simulación de metano con el espín colineal. Notése que en la tercera y cuarta columna los resultados de Mulliken se dividen en aquellos con espín hacia arriba y otros hacia abajo

Por este motivo, el script encargado del tratamiento de datos que habíamos diseñado para recoger el resultado del fichero de salida que emite SIESTA, tuvimos

que modificarlo para que funcionara correctamente según la multiplicidad de espín seleccionada por el usuario. (FIGURA 8)

Además, para que el software sepa cuando tiene que aplicar la búsqueda de espín polarizado colineal, no colineal o no polarizado, fue necesario conectar el árbol de decisión del script de búsqueda, con el tipo de simulación elegida por el usuario. De esta manera, si el usuario ha elegido previamente una simulación de una molécula con espín colineal, al buscar resultados en la sección de análisis, automáticamente el script de búsqueda encontrará los resultados asociados a ese tipo de espín.

3.2.3 Mensajes de error al usuario

Aunque el usuario tiene permitido seleccionar el tipo de multiplicidad de espín con la que efectuar los cálculos de la simulación, no todos ellos permiten el análisis de carga que hemos diseñado. Es el caso del tipo de multiplicidad no colineal y espín-órbita (*spin-orbit*), que no entrega resultados para los métodos Hirshfeld ni Voronoi, por ese motivo, cuando el usuario selecciona este tipo de análisis habiendo activado la multiplicidad espín-órbita, hemos de avisarle de que lo que pide, SIESTA no puede entregárselo. Para ello, programamos una ventana emergente en la que se le informa al usuario antes de la simulación evitando así que el usuario pierda tiempo.

Por otro lado, en caso de que un usuario hiciera caso omiso a los mensajes de advertencia, y forzase al programa a encontrar resultados, hemos modificado el árbol de decisión interno para que, en lugar de activar el script que analiza el fichero de salida de SIESTA muestre un nuevo mensaje de error que le indica al usuario que aún no está implementado dicho análisis en esas condiciones de multiplicidad de espín.

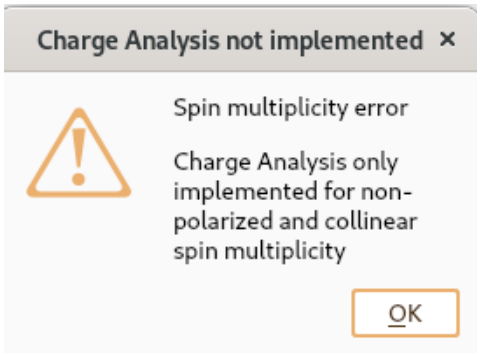


Figura 11. Mensaje de error customizado para el usuario que activa el análisis de cargas tras una simulación con multiplicidad de espín no colineal.

3.2.4 Cambios estéticos

Finalmente, ya que ASAP es un software de simulación de renombre en la comunidad científica, el factor estético que hace agradable el uso del mismo, es vital para proporcionar una buena experiencia de usuario. Características como el ajuste del texto, la expansión de las columnas en función del tamaño de la ventana, alternancia de colores entre filas (para distinguir más claramente entre ellas), etc. Fueron elementos visuales que pese a no guardar relación con el interés científico de este trabajo, sirvieron para culminar completamente mi labor como parte del equipo de desarrollo de software en Simune Atomistics.

CAPÍTULO 4. CONCLUSIONES

Durante una parte de mi estancia en Simune Atomistics, he redactado un tutorial para obtener el análisis de cargas de una sustancia mediante el uso de SIESTA que provee al usuario tanto las instrucciones de activación en el fichero de entrada así como distintos comandos para la extracción de los resultados del fichero de salida.

Por otra lado, como parte del equipo de desarrollo de software de Simune hemos implementado una nueva funcionalidad en ASAP, el análisis de cargas, que permite al usuario elegir entre los métodos de Mulliken Hirshfeld y Voronoi, y obtener los resultados en formato de tabla en una ventana emergente. Hemos diseñado e incluido los test que aseguran el correcto funcionamiento del programa, así como el postprocesado de los datos de Mulliken que permite una comparación más sencilla con los otros métodos. Hemos añadido también la posibilidad de estudio de cargas con diferente multiplicidad de espín así como los mensajes emergentes que informan al usuario de las limitaciones del software.

REFERENCIAS

1. Allen B. Tucker: The Computer Science and Engineering Handbook. CRC Press 1997, ISBN 0-8493-2909-4
2. Bruce W. Arden. What Can Be Automated?: Computer Science and Engineering Research Study (Cosers).
3. Denning, P.J.; Comer, D.E.; Gries, D.; Mulder, M.C.; Tucker, A.; Turner, A.J.; Young, P.R. (1989). "Computing as a discipline"
4. GROMACS 2022.3 Manual. https://doi.org/10.5281/zenodo.7037337.
5. J.M. Yeomans, Mesoscale simulations: Lattice Boltzmann and particle algorithms, Physica A: Statistical Mechanics and its Applications, Volume 369, Issue 1, 2006, Pages 159-184, ISSN 0378-4371, https://doi.org/10.1016/j.physa.2006.04.011.
6. Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E. Castelli, Rune Christensen, Marcin Dułak, Jesper Friis, Michael N. Groves, Bjørk Hammer, Cory Hargus, Eric D. Hermes, Paul C. Jennings, Peter Bjerre Jensen, James Kermode, John R. Kitchin, Esben Leonhard Kolsbjerg, Joseph Kubal, Kristen Kaasbjerg, Steen Lysgaard, Jón Bergmann Maronsson, Tristan Maxson, Thomas Olsen, Lars Pastewka, Andrew Peterson, Carsten Rostgaard, Jakob Schiøtz, Ole Schütt, Mikkel Strange, Kristian S. Thygesen, Tejs Vegge, Lasse Vilhelmsen, Michael Walter, Zhenhua Zeng, Karsten Wedel Jacobsen The Atomic Simulation Environment—A Python library for working with atoms J. Phys.: Condens. Matter Vol. **29** 273002, 2017
7. Artacho, E. , Sánchez-Portal, D. , Ordejón, P. , García, A. and Soler, J. M. (1999), Linear-Scaling ab-initio Calculations for Large and Complex Systems. phys. stat. sol. (b), 215: 809-817. doi:10.1002/(SICI)1521-3951(199909)215:1<809::AID-PSSB809>3.0.CO;2-0 Emilio Artacho, E. Anglada, O. Diéguez, J. D. Gale, A. García, J. Junquera, R. M. Martin, P. Ordejón, J. M. Pruneda, D. Sánchez-Portal and J. M. Soler, The SIESTA method; developments and applicability. Journal of Physics: Condensed Matter, 20 (6), 064208, (2008), url: https://stacks.iop.org/0953-8984/20/i=6/a=064208 José M Soler, Emilio Artacho, Julian D Gale, Alberto García, Javier Junquera, Pablo Ordejón and Daniel Sánchez-Portal, Journal of Physics: Condensed Matter,

14 (11), (2002), url: https://iopscience.iop.org/article/10.1088/0953-8984/14/11/302

8. F. Marchesin, P. Koval,Y. Pouillon, I. Lebedeva, A. García, M. García-Mota, A. Kimmel “Atomistic Simulation Advanced Platform (ASAP) for materials modelling with ab initio methods”, Psi-k conference 2022, Lausanne (Switzerland), abstract book. https://drive.google.com/file/d/1XdqSA-0ZuNsXh4ildQf8VwDsReS4ShZf/view
9. Champagne, B.; Perpète, E. A.; van Gisbergen, S. J. A.; Baerends, E.-J.; Snijders, J. G.; Soubra-Ghaoui, C.; Robins, K. A.; Kirtman, B. Assessment of Conventional Density Functional Schemes for Computing the Polarizabilities and Hyperpolarizabilities of Conjugated Oligomers: An Ab Initio Investigation of Polyacetylene Chains. J. Chem. Phys. 1998, 109, 10489.
10. 2 Abraham, R. J.; Griffiths, L.; Loftus, P. Approaches to Charge Calculations in Molecular Mechanics. J. Comput. Chem. 1982,3, 407-416.
11. Winn, P. J.; Ferenczy, G. G.; Reynolds, C. A.. Toward Improved Force Fields. 1. MultipoleDerived Atomic Charges. J. Phys. Chem. A 1997, 101, 5437-5445.
12. Mobley, D. L.; Dumont, É.; Chodera, J. D.; Dill, K. A. Comparison of Charge Models for Fixed-Charge Force Fields: Small-Molecule Hydration Free Energies in Explicit Solvent. J. Phys. Chem. B 2007, 111, 2242-2254.
13. Sterrer, M.; Risse, T.; Martinez Pozzoni, U.; Giordano, L.; Heyde, M.; Rust, H.-P.; Pacchioni, G.; Freund, H.-J. Control of the Charge State of Metal Atoms on Thin MgO Films. Phys. Rev. Lett. 2007, 98, 096107.
14. Lechtken, A.; Neiss, C.; Stairs, J.; Schooss, D. Comparative Study of the Structures of Copper, Silver, and Gold Icosamers: Influence of Metal Type and Charge State. J. Chem. Phys. 2008, 129, 154304.
15. Tishchenko, O.; Li, R.; Truhlar, D. G. Metal-organic Charge Transfer Can Produce Biradical States and is Mediated by Conical Intersections. Proc. Nat. Acad. Sci. 2010, 107, 19139– 19145.
16. Verma, P.; Xu, X.; Truhlar, D. G. Adsorption on Fe-MOF-74 for C1-C3 Hydrocarbon Separation. J. Phys. Chem. C 2013, 117, 12648–12660.
17. Vilseck, J. Z., Tirado-Rives, J., & Jorgensen, W. L. (2014). Evaluation of CM5 Charges for Condensed-Phase Modeling. J. Chem. Theory Comput. 2014, 10, 2802-2812.

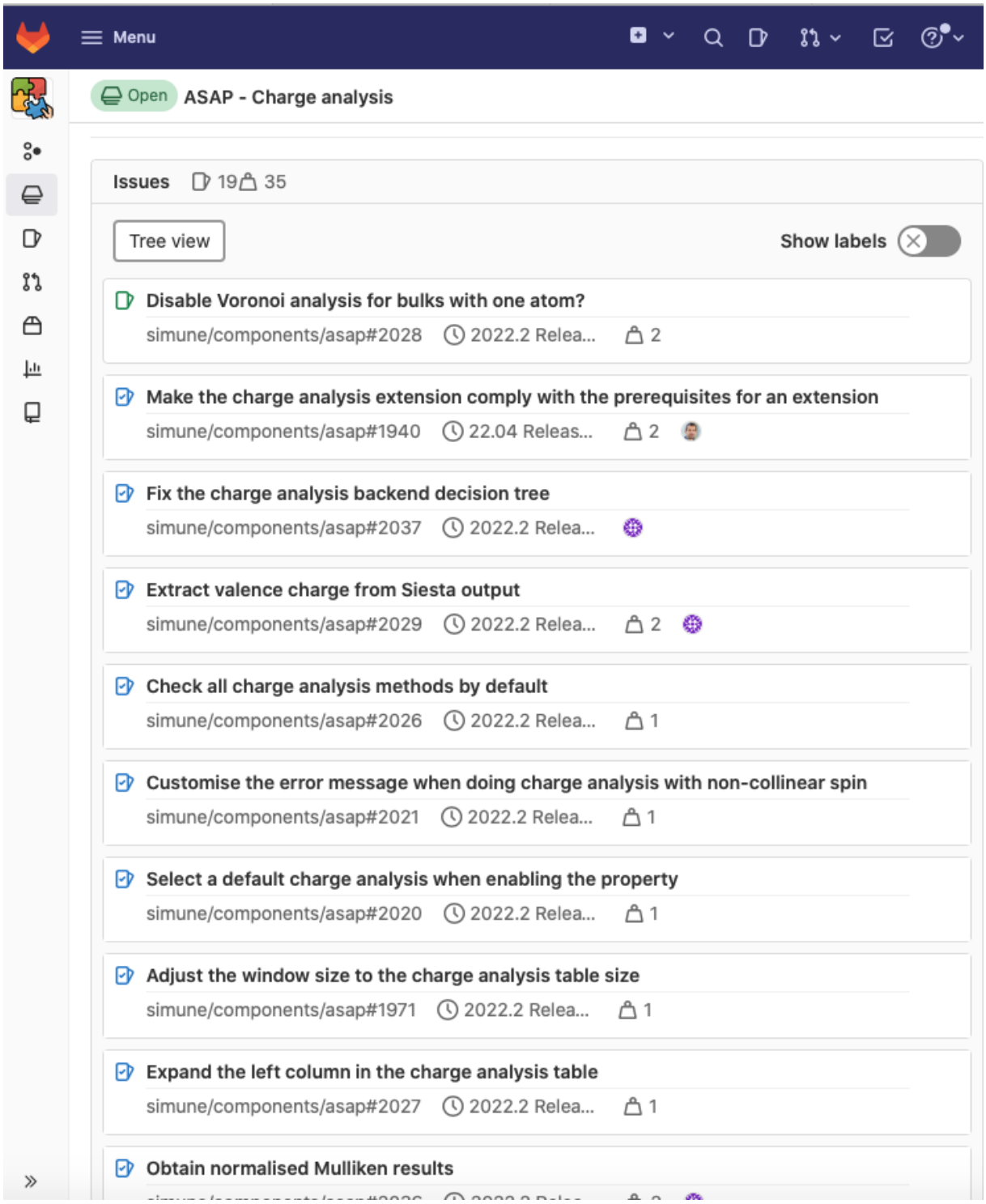
18. Williams, K. S.; Lenhart, J. L.; Andzelm, J. W.; Bandara, S. V.; Baril, N. F.; Henry, N. C.; Tidrow, M. Z. First Principles Investigation of Water Adsorption and Charge Transfer on III-V(110) Semiconductor Surfaces. Surf. Sci. 2014, 622, 71– 82.
19. Unke, O. T.; Meuwly, M. PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges. J. Chem. Theory Comput. 2019, 15, 3678-3693
20. Mulliken, R. S. Electronic Population Analysis on LCAO-MO Molecular Wave Functions I. J. Chem. Phys. 1955, 23, 1833–1840
21. Thompson, J. D.; Xidos, J. D.; Sonbuchner, T. M.; Cramer, C. J.; Truhlar, D. G. More Reliable Partial Atomic Charges When Using Diffuse Basis Sets. PhysChemComm. 2002, 5, 117-134.
22. Henkelman, G.; Arnaldsson, A.; Jónsson, H. A Fast and Robust Algorithm for Bader Decomposition of Charge Density, Comput. Mater. Sci. 2006, 36, 354-360.
23. Sanville, E.; Kenny, S. D.; Smith, R.; Henkelman G. An Improved Grid-Based Algorithm for Bader Charge Allocation. J. Comp. Chem. 2007, 28, 899-908.
24. Tang, W.; Sanville, E.; Henkelman G.; A Grid-Based Bader Analysis Algorithm Without Lattice Bias. J. Phys.: Condens. Matter. 2009, 21, 084204.
25. Hirshfeld F. L. Bonded-Atom Fragments for Describing Molecular Charge Densities. Theor. Chim. Acta. 1977, 44, 129–138.
26. Besler, B. H.; Merz, K. M. Jr.; Kollman, P. A. Atomic Charges Derived from Semiempirical Methods. J. Comp. Chem. 1990, 11, 431-439
27. Wang, B.; Li, S. L.; Truhlar, D. G. Modeling the Partial Atomic Charges in Inorganometallic Molecules and Solids and Charge Redistribution in Lithium-Ion Cathodes. J. Chem. Theory Comput. 2014, 10, 5640-5650.
28. Charge Distribution in the Water Molecule—A Comparison of Methods F. Martin, H. Zipse DOI 10.1002/jcc.20157 J Comput Chem 26: 97–105, 2005
29. I. Choudhuri, Donald G. Truhlar, Calculating and Characterizing the Charge Distributions in Solids, J. Chem. Theory Comput. 2020, 16, 9, 5884–5892
30. Sigfridsson, E.; Ryde, U.Comparison of methods for deriving atomic charges from the electrostatic potential and moments. J Comp Chem 1998, 19, 377.
31. R.S. Mulliken, J. Chem. Phys. 23 (1955) 1833.

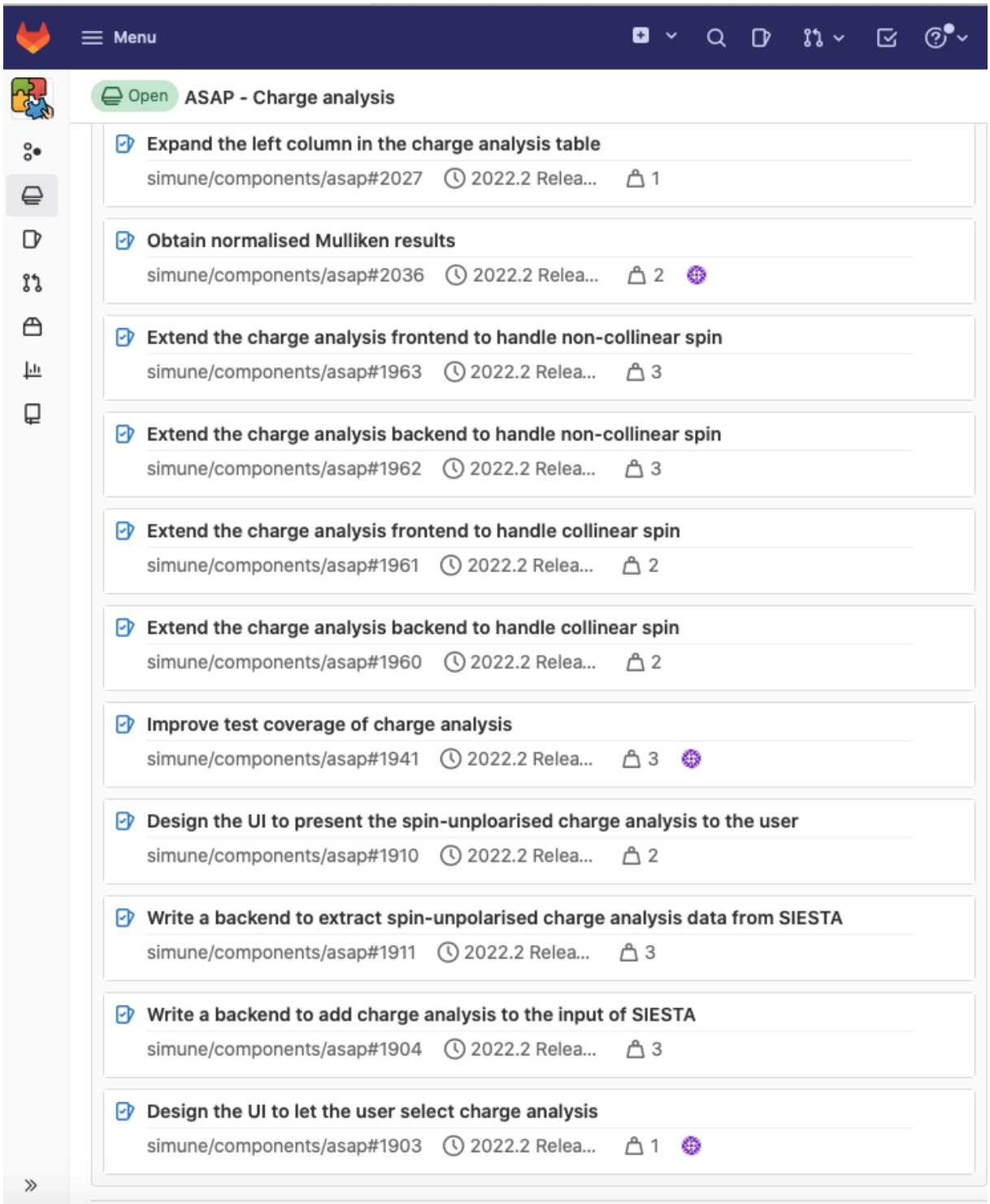
32. R.S. Mulliken, J. Chem. Phys. 23 (1955) 1841.
33. R.S. Mulliken, J. Chem. Phys. 23 (1955) 2338.
34. R.S. Mulliken, J. Chem. Phys. 23 (1955) 2343.
35. Guerra, C. F., Handgraaf, J.-W., Baerends, E. J., & Bickelhaupt, F. M. (2003). Voronoi deformation density (VDD) charges: Assessment of the Mulliken, Bader, Hirshfeld, Weinhold, and VDD methods for charge analysis. Journal of Computational Chemistry, 25(2), 189–210. https://doi.org/10.1002/JCC.10351
36. Voronoi, Georges. "Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire. Sur quelques propriétés des formes quadratiques positives parfaites." Journal für die reine und angewandte Mathematik (Crelles Journal), vol. 1908, no. 133, 1908, pp. 97-102. https://doi.org/10.1515/crll.1908.133.97
37. Voronoi, Georges. "Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les parallélloèdres primitifs." Journal für die reine und angewandte Mathematik (Crelles Journal), vol. 1908, no. 134, 1908, pp. 198-287. https://doi.org/10.1515/crll.1908.134.198
38. Bickelhaupt et al, Organometallics 15, 2923 (1996) and Fonseca et al, J. Comp. Chem. 25, 189 (2003)
39. Bock, Martin; Tyagi, Amit Kumar; Kreft, Jan-Ulrich; Alt, Wolfgang (2009). "Generalized Voronoi Tessellation as a Model of Two-dimensional Cell Tissue Dynamics". Bulletin of Mathematical Biology. 72 (7): 1696–1731. arXiv:0901.4469v1.
40. Löbl, Matthias C.; Zhai, Liang; Jahn, Jan-Philipp; Ritzmann, Julian; Huo, Yongheng; Wieck, Andreas D.; Schmidt, Oliver G.; Ludwig, Arne; Rastelli, Armando; Warburton, Richard J. (2019-10-03). "Correlations between optical properties and Voronoi-cell area of quantum dots". Physical Review B. 100 (15): 155402. arXiv:1902.10145
41. https://departments.icmab.es/leem/siesta/Pseudopotentials/index.html

ANEXO I. Exposición de los *issues* en GitLab

A modo explicativo he incluido en este anexo, dos capturas de pantalla del portal GitLab. En este caso, podemos observar que el Epic ‘’ASAP - Charge Analysis’’ se compone de todos los issues que aparecen en las figuras.

El orden temporal es descendente, de manera que el inicio del trabajo comenzó con los que se encuentran al final.





Al ir resolviendo y cumplimentando cada tarea asignada, iban surgiendo otras nuevas que fui cumplimentando más adelante. Estas dos figuras son un buen resumen paso por paso de mi estancia en SIMUNE Atomistics como parte del equipo de desarrollo de software.

ANEXO II. Resultados de análisis de cargas en sustancias de uso común. Agua, amoníaco y acetonitrilo

A continuación representamos los resultados del análisis de cargas obtenidos mediante el softwqare ASAP de agua, amoníaco y acetonitrilo. Quiero destacar en el resultado final de la composición de la tabla, los márgenes, el orden y la limpieza visual que permite al usuario final estudiar el análisis de cargas de la sustancia deseada. FIgura I.

Charge Analysis: Project 008

Atom index	Symbol	Mulliken	Hirshfeld	Voronoi
1	O	-0.498	-0.309	-0.293
2	H	0.249	0.155	0.146
3	H	0.249	0.155	0.146

Charge Analysis: Project 009

Atom index	Symbol	Mulliken	Hirshfeld	Voronoi
1	N	-0.698	-0.316	-0.296
2	H	0.233	0.105	0.099
3	H	0.233	0.105	0.098
4	H	0.233	0.105	0.098

Charge Analysis: Project 010

Atom index	Symbol	Mulliken	Hirshfeld	Voronoi
1	C	-0.521	-0.063	-0.083
2	C	0.242	0.082	0.069
3	N	-0.369	-0.23	-0.23
4	H	0.216	0.07	0.082
5	H	0.216	0.07	0.081
6	H	0.216	0.07	0.081

Figura I. Resultados del análisis de cargas obtenidos mediante el software ASAP, para las moléculas de agua, (arriba) , amoníaco (centro), acetonitrilo (abajo).