Chess-Dot-Py

Miguel Díaz Hernández

Proyecto de fin de curso
DAM 2 - 2022

Índice

Descripción de proyecto y ámbito de implantación

¿Qué has desarrollado y por qué? ¿Quién lo usará? ¿Qué tecnologías has usado? ¿Se prevén cambios en el futuro?

Descripción del proyecto

Chess.com es uno de los sitios más conocidos y utilizados por jugadores de ajedrez, ya que permite jugar a distintos modos con cualquier persona en el mundo, analizar las partidas y estadísticas de cada jugador y seguir a los jugadores más conocidos en distintas competiciones. También ofrece bots contra los que entrenar y puzzles para resolver.

ChessDotPy es una aplicación de escritorio escrita en Python que usa la API de Chess.com para ofrecer al usuario una forma más simple de visualizar datos, ya que la interfaz de la web puede llegar a ser confusa.

Esta aplicación puede ser utilizada sin necesidad de tener una cuenta creada en la web. Para probarla, recomiendo buscar los perfiles de **Hikaru** y **GothamChess**, ya que son jugadores conocidos en la comunidad y que están afiliados con el propio sitio, así que juegan de forma regular.

Funcionalidades que ofrece

La aplicación permite acceder a una serie de **secciones** mediante la interfaz de usuario:

- **Perfiles de usuario**:
    - Estadísticas de partidas por modo de juego y
    - estadísticas generales de partidas:
        - Puntuación
        - Partidas jugadas
        - Victorias
        - Derrotas
        - Empates
        - Porcentaje de victorias

    - Perfil:
        - Imagen de perfil
        - Nombre de usuario
        - Nombre (no todos los usuarios muestran su nombre real)
        - Seguidores
        - Título (por ejemplo GM - Grandmaster)
        - Fecha de registro
        - Fecha de última conexión
        - Estado (premium o no premium)

- **Historial de partidas de usuario**:
    - Color de las piezas
    - Resultado de la partida
    - Precisión de los movimientos
    - Puntuación del jugador al empezar la partida
    - Formato de tiempo de la partida
    - Normas utilizadas en la partida
    - Fecha y hora en la que empezó la partida

- **Tabla de clasificación de distintos modos de juego**:
    - Foto de perfil
    - Nombre de usuario
    - Nombre
    - Puntuación
    - Estadísticas (victorias, derrotas, empates)
    - País
    - Flair

- **Rompecabezas diario y aleatorio**:
    - Datos (no todos tienen los mismos datos)
        - Evento
        - Localización
        - Fecha
        - Ronda
        - Jugadores
        - Resultado
    - Permite *mostrar la solución*, funcionalidad que no está presente en la web

El uso y navegación de la interfaz es simple e intuitivo. Incluye una serie de shortcuts de teclado (incluídos en el README) para realizar diferentes acciones sin tener que clicar directamente en los botones.

Tecnologías usadas

Para crear esta aplicación, se utilizaron diferentes tecnologías:

- Python
- Qt5
- Pyinstaller
- Github Actions

Se utilizó **Python** debido a su **facilidad de uso** y **versatilidad**, combinado con Qt5 (específicamente, una versión para Python llamada **PyQt5**) para crear las **interfaces de usuario**, y con **Github Actions** para **control de versiones** y **generación de ejecutables automática**, con ayuda de **Pyinstaller**.

Puede que en un futuro, si se actualiza la API de Chess.com, sea necesario actualizar la aplicación o las dependencias que utiliza.

Recursos de hardware y software

¿Qué requisitos se necesitan para probar el proyecto (procesador, memoria, sistema operativo...)? ¿Has empleado algún IDE o complemento en especial? ¿Dispones de entorno de desarrollo y entorno de producción diferenciados?

Requisitos necesarios

Python: La versión de Python utilizada es la 3.10.4.

Librerías: Todas las librerías requeridas se descargan con sus respectivas versiones siguiendo la guía de instalación en el repositorio.

En caso de estar utilizando el ejecutable autogenerado por las Github Actions, no es necesario instalar ni Python ni las librerías, ya que vienen incluídas.

Sistema operativo:

- Utilizando el ejecutable: Probado Windows 10 y Windows 11 (64 bits)
- Python: Cualquier sistema operativo que soporte la versión de Python usada

Hardware: No es necesario ningún hardware específico, ya que el consumo de RAM de la aplicación es mínimo (normalmente utiliza de 40mb a 150mb, si todas las fotos de perfil están cargadas) y funciona con gráficos integrados.

Red: Es necesaria una conexión de red. Las operaciones que más capacidad de red consumen (descarga de fotos de perfil en la tabla de clasificación, por ejemplo) se ejecutan en segundo plano una vez los datos básicos se cargan.

Software utilizado

El IDE empleado ha sido **Visual Studio Code**, ya que se adapta a una amplia gama de lenguajes y soporta una variedad de plugins, customización y herramientas que facilitan mucho el desarrollo.

No utilicé ninguna extensión aparte de las recomendadas para utilizar Python, pero sí que cambié los ajustes de formato para que se aplique automáticamente al guardar los archivos.

Esta funcionalidad es útil porque controla automáticamente elementos como la longitud de las líneas, espacios entre funciones, etc.

Entorno

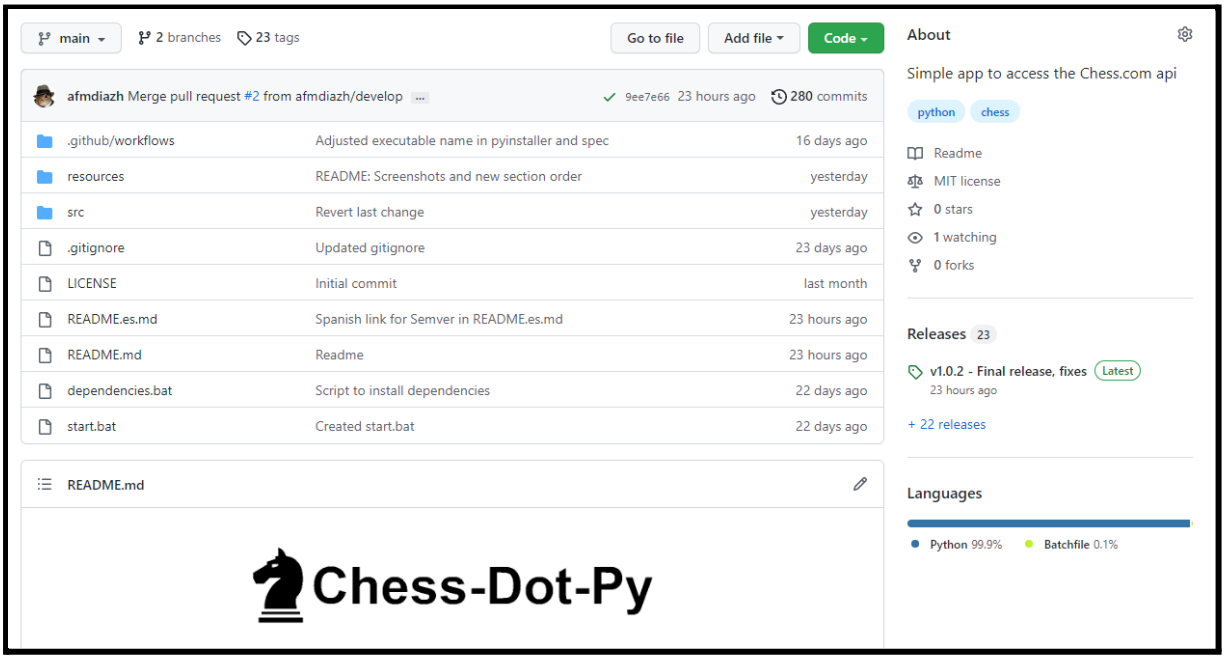
En este caso, el entorno de desarrollo ha sido mi propio ordenador, y el entorno de almacenamiento del código y de producción ha sido Github, gracias a las **Github Actions**.

Éstas se encargan de generar **ejecutables** y **versiones** de forma **automática** con cada tag añadido (un tag sería v1.2.3, por ejemplo).

Dentro de Github, usé diferentes branches:

- **main**: rama principal, almacena solo código estable.
- **develop**: rama de desarrollo.
- **readme**: rama destinada a modificar el readme.

El uso de branches no es necesario ya que soy la única persona trabajando en el proyecto, pero con el propósito de mantener el repositorio limpio las utilicé de todas formas.



La carpeta .github/workflows contiene las actions, que se ejecutan cuando se cumplen condiciones específicas (puede ser en cada push, pull request, cuando lo solicite un usuario…), en este caso, cada vez que se crea un tag con el formato vX.X.X.

La sección Actions contiene todas las ejecuciones de estas acciones, y la sección Releases contiene todas las versiones generadas automáticamente con un pequeño changelog de todos los commits realizados.

Los ejecutables presentes en la sección de Releases son los mismos presentes en la pestaña de Actions, pero añadidos manualmente a cada versión. El funcionamiento de éstos está comprobado.

Temporalización del proyecto y fases del desarrollo

¿Qué planificación has seguido para llevar a cabo el proyecto? ¿Puedes plasmarla en diagramas Gantt/PERT? ¿Con qué retos te has enfrentado? ¿Qué has aprendido?

Planificación

No utilicé ninguna planificación en específico, pero mi proyecto se divide en distintas fases.

Fases

1. Idea de proyecto (2 abr - 17 abr)

Lo primero que hice fue pensar en qué idea quería seguir para el proyecto. Estaba seguro de que quería utilizar una API porque durante el curso nos centramos en **utilizar APIs para obtener datos de sitios externos**.

Tras bastante navegación por un repositorio de apis públicas y diferentes ideas descartadas, decidí hacer un proyecto sobre Chess.com, ya que es una web que había utilizado anteriormente y con la que estoy familiarizado.

Este paso llevó tiempo, ya que **muchas APIs fueron descartadas** por limitaciones que tenían (límite de peticiones por hora, límite de uso o capacidad sin comprar una licencia, etc.).

Algunas de las ideas descartadas fueron:

- Aplicación para generar código mediante Inteligencia Artificial con OpenAI.
- Aplicación para traducir a diferentes idiomas ficticios con Fun Translations.
- Aplicación para mostrar información de personajes de Disney con Disney API.

2. Selección de entorno (18 abr)

El siguiente paso fue seleccionar el entorno (aunque ya tenía la idea de hacerlo en Python). Decidí usar **Python**, ya que lo cubrimos de forma superficial durante el curso y me gustaría profundizar algo más.

Para alojar el proyecto, decidí usar **Github** ya que, aunque no tiene las funcionalidades de visualización del proyecto que tiene Gitlab, tiene las **Github Actions** que serán de gran ayuda a la hora de crear versiones y releases de forma automática.

3. Estructura del proyecto (18 abr)

Este paso puede que sea el más importante, ya que tener una estructura del proyecto definida ayudará a que sea más fácil de entender, programar y expandir en caso de que sea necesario.

La estructura del proyecto es bastante sencilla. Los datos se obtienen de la API de Chess.com y se almacenan utilizando clases que representan los distintos elementos de las respuestas obtenidas de la API. Estas clases facilitan el acceso y representación de los datos, además de que permiten crear métodos para procesarlos.

La interfaz obtendrá los datos almacenados en estas clases para mostrarselos al usuario, cuando éste las pida mediante interacciones con la propia interfaz.

4. Desarrollo del proyecto (19 abr - 25 may)

Para la programación me centré en **desarrollar funcionalidades básicas** e ir expandiendo, para poder entregar un proyecto funcional en caso de que no me diese tiempo a implementar todo lo que tenía pensado.

Empecé con los esqueletos de funcionalidades básicas (**DTOs para jugadores**, **tabla de clasificación**, el esqueleto de la **interfaz**) y a partir de ahí me centré en mejorar la funcionalidad y añadir nuevos elementos.

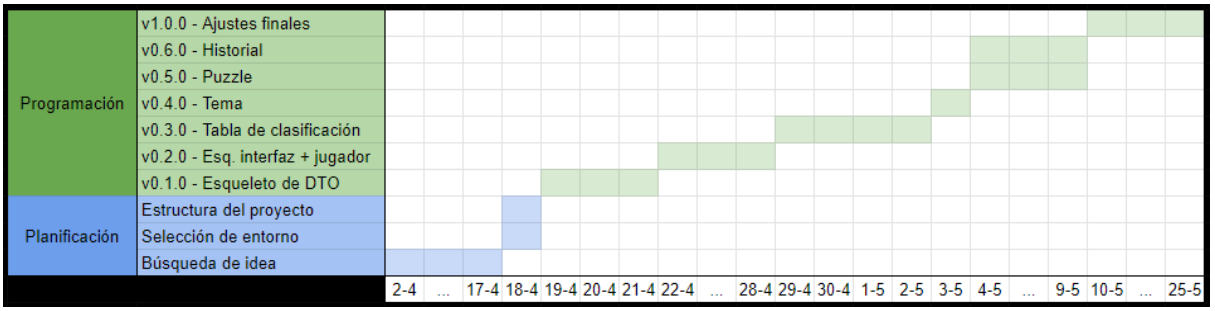
Las distintas **fases de desarrollo** fueron:

1. Crear esqueleto de clases
2. Crear esqueleto de interfaz
3. Comunicación entre la interfaz y los datos
4. Generación de ejecutables automática
5. Implementación de nuevas funcionalidades
6. Arreglo de bugs y ajustes finales

Para la implementación de nuevas funcionalidades me adapté según el tiempo que tenía. Lo primero que implementé fue el perfil y la tabla de clasificación. Una vez terminadas esas secciones añadí el puzzle diario e historial. Luego me dediqué a las modificaciones finales y arreglo de errores.

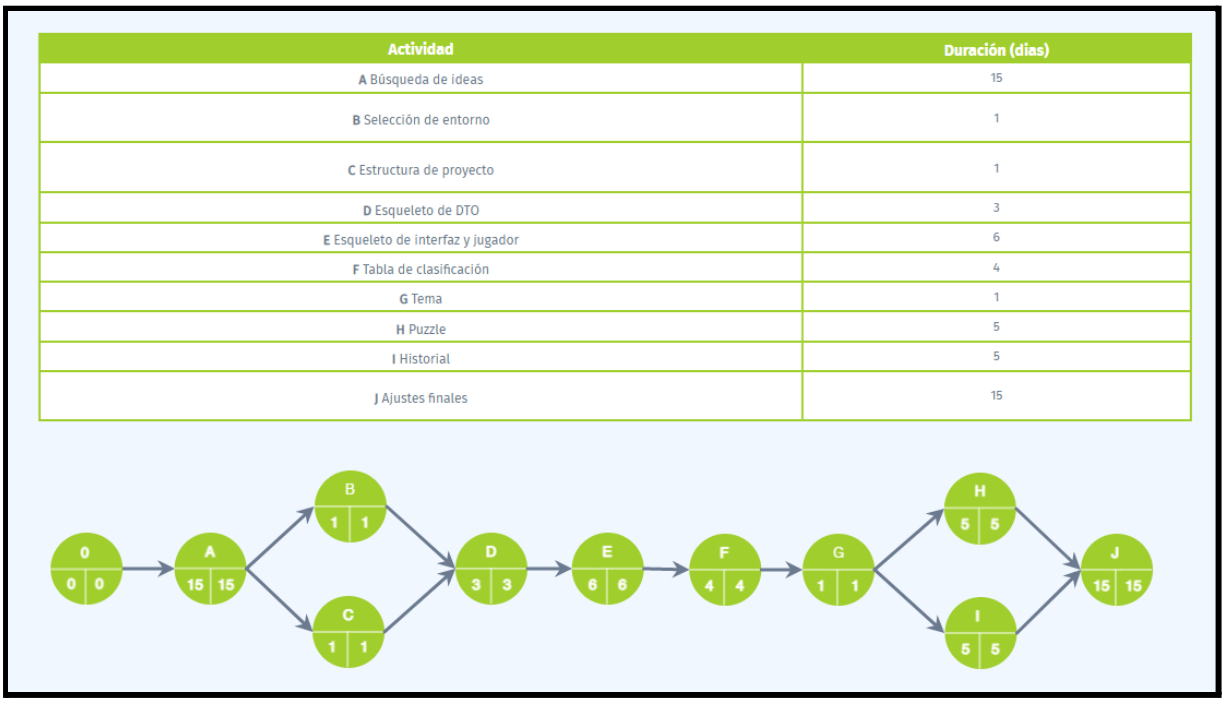
Diagramas

Gantt



(los puntos suspensivos representan varios días)

PERT



Actividad	Duración (días)
A Búsqueda de ideas	15
B Selección de entorno	1
C Estructura de proyecto	1
D Esqueleto de DTO	3
E Esqueleto de interfaz y jugador	6
F Tabla de clasificación	4
G Tema	1
H Puzzle	5
I Historial	5
J Ajustes finales	15

Algunos puntos no son representados de forma adecuada en los diagramas, ya que el desarrollo no fue continuo, hubo períodos de tiempo entre distintas versiones en los que no estuve trabajando en la aplicación.

Por ejemplo, el período de los ajustes finales son 15 días, pero casi todos los cambios fueron hechos el día 25.

Retos

Encontrar una API sin límites: Antes de seleccionar la idea de Chess.com, tenía en mente otras APIs que se podrían utilizar para crear un proyecto interesante.

Por ejemplo, una API de traducción o otra de Inteligencia Artificial para generar texto a partir de un input del usuario.

Por desgracia, la mayoría de las APIs públicas tiene limitaciones, o bien de cantidad de peticiones que puedes hacer o son de pago.

Adaptación a nuevas herramientas: Utilicé una librería para la interfaz diferente a la que utilizamos en clase cuando aprendimos Python, así que adaptarme a ella llevó un tiempo.

Optimización de descarga de imágenes: La descarga de imágenes fué un reto en especial porque la descarga en segundo plano utilizando requests no parecía funcionar desde la clase de hilos que proporciona PyQt5, por lo que la interfaz se congelaba al descargar todas las imágenes de la tabla de clasificación.

La solución fué usar los propios hilos de Python, que suelen ser compatibles con PyQt5 siempre que no se creen elementos de la interfaz desde éstos.

Otro ajuste que mejoró el rendimiento de las descargas fué, en lugar de crear un hilo para cada descarga individual de una imagen de perfil, agruparlas por tablas, porque iniciar los hilos también causa un pequeño impacto de rendimiento.

Subclases de elementos de interfaz: Como PyQt5 no implementa ninguna funcionalidad para mantener el aspect ratio de las imágenes, fué necesario crear una subclase en la que implementé código para que se mantuvieran las dimensiones y la imagen no se viera distorsionada al cambiar el tamaño de la ventana.

También fué necesaria la implementación de una subclase de tablas para almacenar los datos de cada entrada individualmente, para la pestaña de detalles de cada tabla.

Aprendizaje

Realizando este proyecto aprendí a:

- Utilizar las librerías de Python Chess.com y PyQt5.
- Utilizar el archivo README.md para mostrar información sobre el repositorio.
- Utilizar las Github Actions para crear versiones y ejecutables automáticamente.

Descripción de datos

Las partes significativas del proyecto... ¿Cómo funcionan? ¿Puedes elegir ciertos procesos o funciones que sean clave y detallarlos? ¿Qué tipos de datos se manejan (entrada, intermedios, salida)? ¿Cómo?

Procesos clave

Descarga de datos

La funcionalidad principal en la que se basa el proyecto es la librería Chess.com, que se comunica con la API de la página.

Mediante la librería hacemos peticiones a la página solicitando datos específicos, por ejemplo perfiles de usuarios específicos o tablas de clasificación. Estos datos se reciben como Chess.com response y se convierten a json.

Los datos recibidos en formato json se utilizan para instanciar objetos de las clases que representan esos tipos de datos. Por ejemplo, una respuesta de la tabla de clasificación se convertirá en un objeto Leaderboard, que contendrá LSections (leaderboard sections) y a la vez estos contendrán LPlayers (leaderboard players).

Estas funciones de descarga de datos se ejecutan en segundo plano para que no interrumpa la ejecución de la ventana en el hilo principal, por lo que el usuario siempre puede utilizar el programa, aunque los datos se estén descargando.

Interfaz

La interfaz es lo primero que se ejecuta al iniciar el programa, y se actualiza mediante eventos. Por ejemplo, cuando el usuario pulsa el botón para buscar un jugador, se ejecuta un evento que lanza los hilos para solicitar los datos.

Estos hilos están enlazados a funciones, que reciben los datos descargados y se ejecutan en el hilo principal una vez las descargas acaban.

Estas funciones se encargan de utilizar los datos de las clases y representarlos en la interfaz.

Actions

Este proceso no es parte de la aplicación en sí, pero es importante. Las actions se encuentran dentro de la carpeta .github/workflows y se ejecutan en ciertos eventos

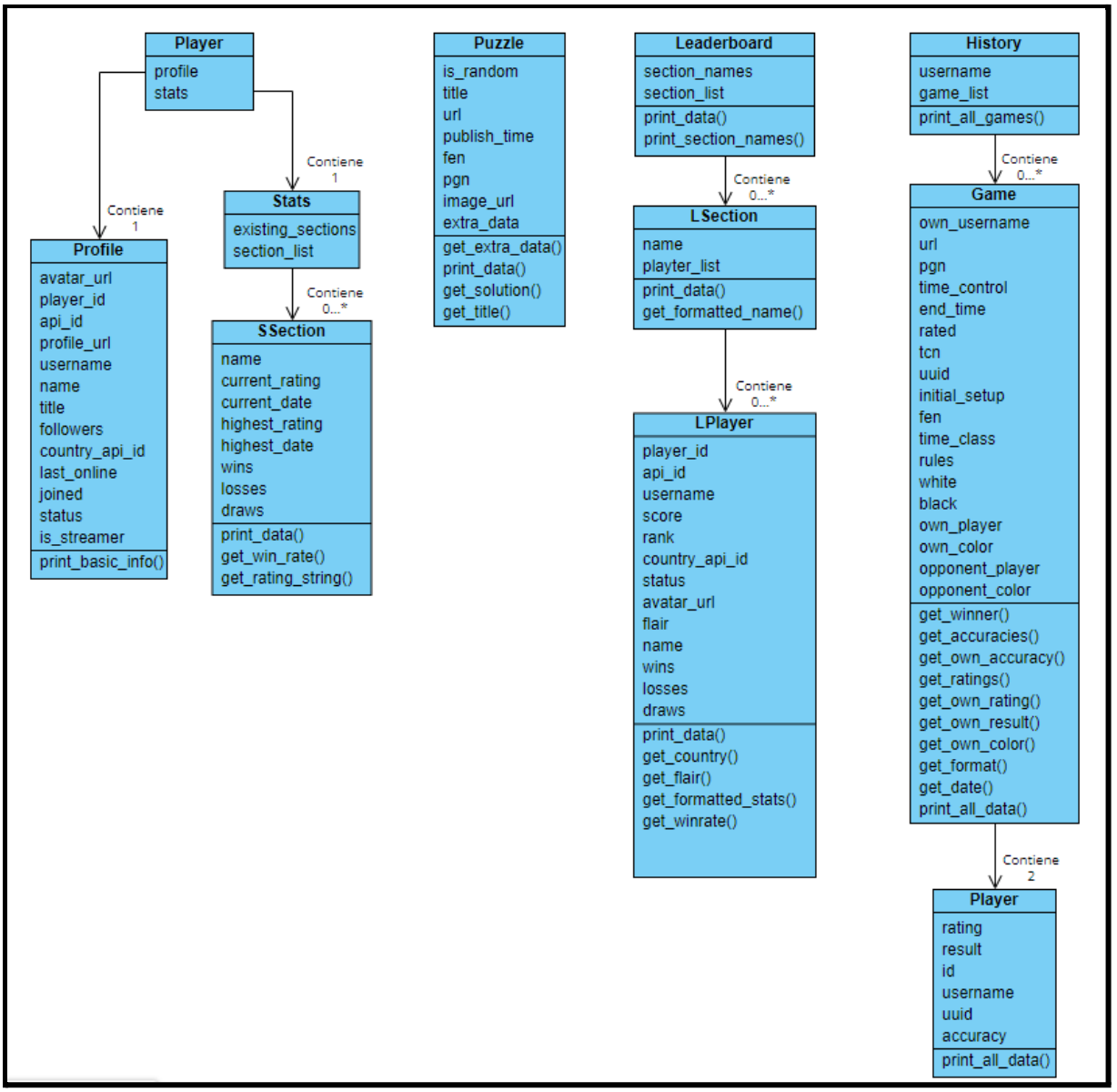
Usando Pyinstaller y el archivo main.spec, las actions generan un ejecutable de forma automática en cada versión.

Arquitectura software y de aplicación

¿Puedes detallar en UML los componentes software del proyecto y su interrelación? ¿Has aplicado algún patrón de diseño o arquitectura? Y dejando a un lado el código... ¿Cómo se explica a alto nivel el funcionamiento de la aplicación? ¿Qué actores (humanos, dispositivos, BBDD, ...) hay involucrados y cómo están conectados? ¿Se usan protocolos o estándares específicos en esa comunicación?

Clases

Diagrama UML de las clases utilizadas para almacenar las distintas respuestas.



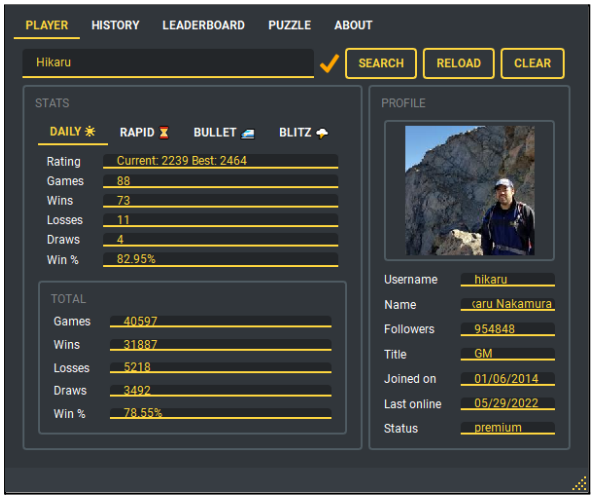
Funcionamiento

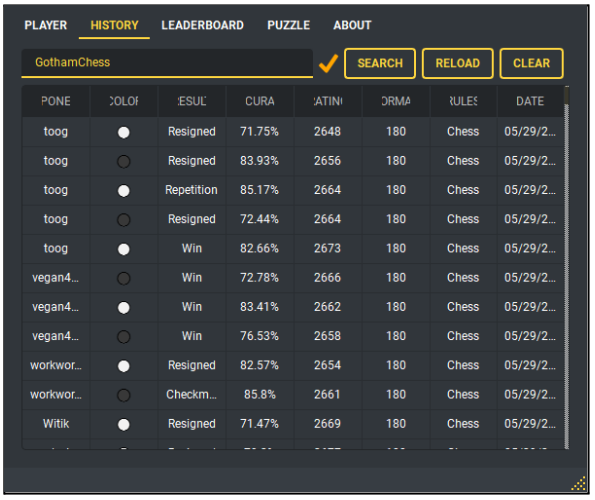
El funcionamiento de la aplicación es simple:

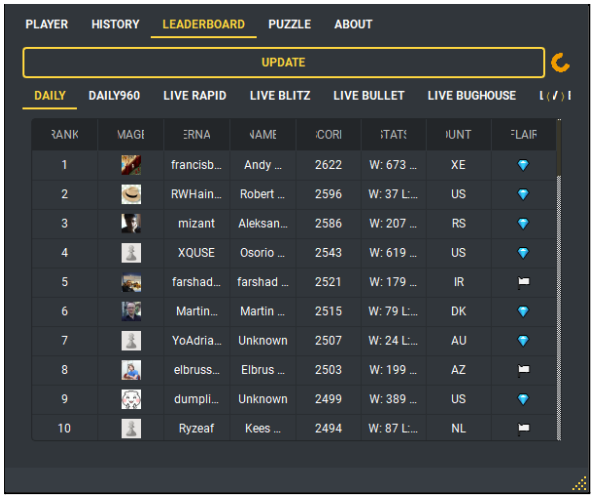
1. Se carga la interfaz.
2. Se conectan los eventos de la interfaz con funciones.
3. El usuario interactúa con la interfaz.
4. Las funciones lanzan los hilos para obtener los datos.
5. Una vez descargados los datos, se actualiza la interfaz.

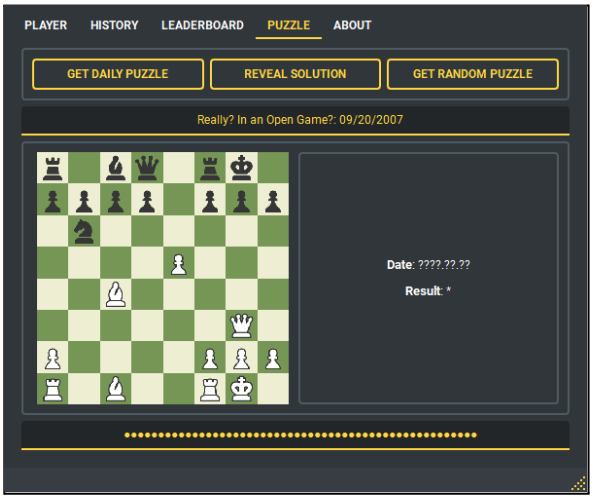
Capturas de pantalla

Algunas capturas de pantalla de cómo se ve el programa en Windows 10.









Conclusión

Mi conclusión al acabar el proyecto es que, aunque Python no está diseñado para este tipo de aplicaciones, su simpleza y facilidad de uso hace que sea muy útil para aplicaciones pequeñas.

Como parte negativa diría que, al ser un lenguaje interpretado, el rendimiento es mucho peor que cualquier otro lenguaje compilado (aunque en un proyecto pequeño como éste la diferencia es casi inexistente), y se dificulta mucho la creación de archivos ejecutables y la protección del código.

Los ejecutables deben ser generados con una librería externa, y son de gran tamaño ya que deben incluir Python y todas las dependencias necesarias.

En cuanto a la plataforma de destino, creo que crear una aplicación para escritorio fue una buena idea ya que durante el curso no profundizamos mucho en ello, ya que nos centramos más en Android, y facilita probar la aplicación.

Y en cuanto a funcionalidad, me gustaría haber añadido alguna otra estadística para los perfiles de jugador, pero por desgracia, la API sólo proporciona datos básicos: victorias, derrotas y empates. Puede que en un futuro se añadan otros elementos, como openings más utilizados, porcentaje de victoria con cada opening, con cada color de piezas… etc.