

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Integrantes del grupo: Juan Sebastián Rodríguez Carvajal, Adrián David Espitia Bayona.

ChessAnalist

Propósito.....	2
Requerimientos Funcionales.....	2
Requerimientos no funcionales.....	3
Clases en java.....	4
Clases encargadas de la autenticación:.....	4
FirestoreServices (Modelo) : En esta clase se configura la conexión a la base de datos y se definen métodos necesarios para la autenticación y el registro de los usuarios al sistema. Resaltar que esta clase es totalmente estática, se hace de esta manera al considerar que esta clase no depende de instancias específicas.....	4
AuthController (Vista-modelo): Esta clase se encarga de la lógica en la verificación de datos que el usuario ingresa al iniciar sesión ó registrarse, tales como que el nombre tenga como mínimo cierta cantidad de caracteres y no sobrepase otra cantidad. Es importante resaltar que AuthController primero verifica los datos que ingresa el usuario antes de comunicar con Firestore.....	5
AuthView (Vista): Esta es la vista de autenticación, en la cual se definen los diseños de la interfaz de usuario.....	6
Clases encargadas del siguiente caso de uso: Análisis de juegos.....	7
GamesFromPgn (Modelo): Esta clase permite obtener una lista de juegos a partir de la ruta a un archivo .png o alternativamente, es posible instanciarla con una lista de juegos cuando los juegos no se obtienen de un archivo .pgn sino desde lichess.org.....	7
OpeningHistogram (Modelo): Esta clase permite anotar cada juego con su respectiva apertura y luego producir una gráfica de barras con el número de veces que dicha apertura fue jugada y la proporción de victorias de cada bando.....	7
GameAnalysis (Modelo): Esta clase ejecuta el análisis de cada una de las posiciones de un juego, detecta jugadas que producen grandes saltos en la evaluación del juego y guarda los resultados para ser usados por otras clases.....	7
Clases encargadas del siguiente caso de uso: Ejercicios de ajedrez.....	8
StockFish (Modelo): Esta clase se encarga de conectar con el motor de ajedrez local para pedirle las evaluaciones de las posiciones.....	8
ChessExercises (Modelo): Clase para instanciar los ejercicios de ajedrez:.....	9
OutPuts (Modelo): En esta clase se definen mensajes predeterminados para el usuario. Se definen los mensajes para cuando el usuario hace un buen movimiento o un mal movimiento. Ejemplo: “¡Gran movimiento! sigue así”.....	10
ChessExercisesController (Vista-modelo): Esta clase se encarga de interactuar con el proceso de StockFish. Obtiene las evaluaciones y maneja la lógica necesaria para determinar si determinado movimiento de pieza fue bueno o no y de tal manera informar al usuario.....	11
Clases encargadas del siguiente caso de uso: Recursos de ajedrez.....	12
ChessResource (Modelo): Clase para instanciar un recurso de ajedrez, ejemplo de recursos: videos de youtube, libros y artículos.....	12
ChessResourceModel (Modelo): En esta clase se instancian los recursos que se van a presentar al usuario.....	13
ChessResourcesController (Vista-modelo): Esta clase se comunica con la vista para desplegar los recursos.....	14
ChessResourcesView (Vista): Esta es la vista de los recursos de ajedrez.....	15
Manual de Usuario.....	17
Menú principal.....	17
Cargar Juegos desde Lichess.org.....	18
Cargar Juegos desde un archivo .png.....	19
Analizar un juego.....	20

Propósito

-Este documento tiene como objetivo presentar el diseño y desarrollo de una aplicación enfocada en el estudio y mejora del rendimiento en ajedrez. La aplicación integra tres casos de uso principales: análisis de partidas de ajedrez mediante el motor Stockfish, resolución de ejercicios tácticos para el desarrollo de habilidades estratégicas y la consulta de recursos especializados, como aperturas, finales y teoría ajedrecística. A través de estas funcionalidades, se busca proporcionar una herramienta completa y eficiente para el aprendizaje y perfeccionamiento del juego.

Requerimientos Funcionales

Autenticación: El usuario deberá poder registrarse e iniciar sesión en el sistema mediante la GUI, ingresando un nombre y contraseña válidos. El usuario podrá realizar las siguientes acciones:

- Ingresar nombre y contraseña.
- Registrarse en el sistema.
- Iniciar sesión.

Análisis de partidas: El usuario deberá poder ingresar su usuario de la plataforma lichess para descargar las partidas desde la misma y analizarlas. El usuario podrá realizar las siguientes acciones:

- Ingresar su usuario de lichess.
- Analizar las partidas descargadas.
- Visualizar sus errores y mejoras.
- Visualizar puntos fuertes en su juego.
- Visualizar estadísticas sobre sus aperturas y el respectivo porcentaje de victorias.

Ejercicios de ajedrez: El usuario podrá seleccionar ejercicios de ajedrez clasificados en: táctica, finales y mates; donde se despliega cada posición del ejercicio en el tablero de ajedrez. El usuario podrá hacer las siguientes acciones:

- Realizar una jugada desde el tablero y recibir retroalimentación sobre la calidad de la jugada hecha.
- Pedirle a la máquina que realice una jugada.
- Retroceder en el ejercicio a la posición anterior.
- Reiniciar el ejercicio desde el principio.

- Pasar al siguiente ejercicio.
- Hacer un análisis libre de una posición configurada por el usuario.

Recursos de ajedrez: El usuario podrá ver en otra vista de la aplicación, una lista de recursos de ajedrez clasificados por la fase del juego (Apertura, medio juego y final) con un nombre, una pequeña descripción y un link que los redirija al recurso en cuestión, el usuario podrá hacer las siguientes acciones:

- Visualizar los recursos de ajedrez propuestos.
- Dirigirse a cada recurso.

Requerimientos no funcionales

Eficiencia: El usuario deberá poder:

- Recibir una respuesta del sistema a cualquier acción en el caso de uso de los ejercicios de ajedrez en menos de 5 segundos.
- Recibir la respuesta del sistema en menos de 1 minuto cuando se trata de analizar las partidas descargadas.
- Recibir respuesta del sistema en menos de 10 segundos para las acciones de autenticación y registro.
- Hacer el cambio entre vistas en menos de 3 segundos.

GUI: La interfaz de usuario deberá cumplir con lo siguiente:

- Manejar una paleta de colores adecuada.
- Manejar una coherencia de estilos entre todas las vistas disponibles.
- Manejar un buen estilo de letra y un buen tamaño de letra.

Clases en java

-Antes de comenzar con la explicación de las clases, se debe decir la arquitectura de software utilizada para el proyecto: MVVM, en el cual se separó las clases en tres carpetas: modelo, vista-modelo y vista. Para la carpeta “modelo” se anexaron aquellas clases que proporcionaban datos requeridos para el funcionamiento del proyecto, tales como los servicios de firebase y las evaluaciones del programa de ajedrez Stockfish. Para la carpeta de “vista-modelo” se anexaron las clases relacionadas con la lógica del manejo y control de datos, tales como el controlador de la evaluación y el controlador de autenticación. Y finalmente, para la carpeta de “vista” se anexaron las clases relacionadas con la interfaz de usuario GUI.

Clases encargadas de la autenticación:

FirestoreServices (Modelo) : En esta clase se configura la conexión a la base de datos y se definen métodos necesarios para la autenticación y el registro de los usuarios al sistema. Resaltar que esta clase es totalmente estática, se hace de esta manera al considerar que esta clase no depende de instancias específicas.

➤ atributos de FirestoreServices:

- FirebaseDatabase: en este atributo se guarda la referencia a la base de datos firebase para manejar las acciones sobre la misma.

➤ Métodos de FirestoreServices:

- InitFirestore(): Este método se utiliza para inicializar la conexión con la base de datos realtime de Firestore.
- PushUser(): Este método se utiliza para registrar un usuario en la base de datos de Firestore. Se registra nombre del usuario y contraseña.
- GetUser(): este método define la forma como se recupera la información del usuario de la base de datos Firestore. Se va a recuperar el nombre del usuario y su contraseña.

AuthController (Vista-modelo): Esta clase se encarga de la lógica en la verificación de datos que el usuario ingresa al iniciar sesión o registrarse, tales como que el nombre tenga como mínimo cierta cantidad de caracteres y no sobrepase otra cantidad. Es importante resaltar que AuthController primero verifica los datos que ingresa el usuario antes de comunicar con Firebase.

➤ Atributos AuthController:

- ValidateName: Esta variable guardará un mensaje de error en caso tal que la verificación del nombre no haya sido exitosa, ejemplo: "El nombre de usuario debe contener como mínimo 4 caracteres".
- ValidatePassword: Esta variable guardará un mensaje de error si la contraseña que el usuario ingresa no es válida, ejemplo: "La contraseña debe tener como mínimo 7 caracteres".

➤ Métodos de AuthController:

- LoginUser(): Maneja toda la lógica para comprobar la información del usuario antes de hacer cualquier solicitud a Firebase.
- RegisterUser(): Maneja la lógica para registrar un usuario. Verifica primero si la información que ingresa el usuario es válida para registrar en el sistema.

```
public class AuthController {
    private static String validateName;
    private static String validatePassword;
    public static void loginUser(String userName, String password, AuthView view) {
        if (!validateName(userName)) {
            view.showErrorMessage(validateName);
            return;
        } else if (!validatePassword(password)) {
            view.showErrorMessage(validatePassword);
            return;
        }
        view.showLoadingMessage(0);
        new Thread(new Runnable() {
            @Override
            public void run() {
                // Simulando el proceso de Firebase (puedes reemplazarlo por Firebase)
                try {
                    Thread.sleep(20);
                    User user = FirebaseServices.getUser(userName, view);
                    if (user == null) {
                        return;
                    }
                    if (user.getPassword().equals(password)) {
                        System.out.println("contraseña correcta");
                        view.dispose();
                    }
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }
}
```

AuthView (Vista): Esta es la vista de autenticación, en la cual se definen los diseños de la interfaz de usuario.

➤ Atributos de AuthView:

- LoginButton: Botón para iniciar sesión.
- RegisterButton: Botón para registrarse.
- PasswordField: Campo para ingresar la contraseña.
- NameField: Campo para ingresar el nombre.

➤ Métodos de AuthView:

- AuthView(): Constructor de la vista.

```
public class AuthView extends JFrame {
    private JButton loginButton;
    private JButton registerButton;
    private JPasswordField passwordField;
    private JTextField nameField;
    private JLabel outPut;
    private boolean isLoading;
    private int type;

    public AuthView() {
        setTitle("Authentication");
        setSize(600, 470);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        getContentPane().setBackground(new Color(213, 249, 222));
        setLayout(new BorderLayout());

        outPut=new JLabel("<html>"+<span style='font-size:13pt'>"+";A divertirse
        outPut.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
        JPanel messagePanel=new JPanel();
        messagePanel.setBackground(new Color(213, 249, 222));
        messagePanel.setBorder(BorderFactory.createEmptyBorder(60, 10, 0, 10));
        JLabel messageLabel = new JLabel("Bienvenido, por favor regístrate o inic:
        messageLabel.setFont(new Font("Arial", Font.BOLD, 14));
        messagePanel.add(messageLabel);
        add(messagePanel, BorderLayout.NORTH);
```

Clases encargadas del siguiente caso de uso: Análisis de juegos.

GamesFromPgn (Modelo): Esta clase permite obtener una lista de juegos a partir de la ruta a un archivo .png o alternatively, es posible instanciarla con una lista de juegos cuando los juegos no se obtienen de un archivo .pgn sino desde lichess.org

➤ Atributos de GamesFromPgn:

- pgn: Un objeto de tipo PgnHolder que procesa y extrae los juegos del archivo .pgn
- games: La lista de juegos.
- headerGameMap: Un objeto de tipo Hashmap que asocia un encabezado con los nombres de los contrincantes y el resultado final a cada juego.

➤ Métodos de GamesFromPgn:

- GamesFromPgn(): Dos constructores, uno que tiene como argumento la ruta al archivo .pgn y otro que toma una lista de juegos como argumento.
- GetGames(): Devuelve la lista de juegos en el atributo games
- getHeaderMap(): Devuelve el Hashmap en el atributo headerGameMap.

OpeningHistogram (Modelo): Esta clase permite anotar cada juego con su respectiva apertura y luego producir una grafica de barras con el número de veces que dicha apertura fue jugada y la proporción de victorias de cada bando.

➤ Atributos de OpeningHistogram:

- chart: Un objeto de tipo JFreeChart que representa el gráfico de barras.

➤ Métodos de OpeningHistogram:

- OpeningHistogram(): Un constructor que toma como argumento una lista de juegos List<Game>, anota cada juego con su respectiva apertura, crea una tabla con la cuenta de victorias y derrotas por cada bando para cada apertura y produce el gráfico de barras con dicha información
- getChart(): Devuelve el gráfico de barras.

GameAnalysis (Modelo): Esta clase ejecuta el análisis de cada una de las posiciones de un juego, detecta jugadas que producen grandes saltos en la evaluación del juego y guarda los resultados para ser usados por otras clases

➤ Atributos de GameAnalysis:

- game: Un objeto de tipo Game que tiene el juego a ser analizado.

- user: El nombre de usuario el cuál se usa para identificar el color de las piezas con las que jugó el usuario en este juego en particular.
- moves: La lista de movimientos del juego.
- fenList: La lista de posiciones en notación fen.
- moveEvaluations: Un Hashmap que a cada movimiento le asocia la correspondiente evaluación de la posición resultante.
- deltaEvaluations: Un Hashmap que a cada movimiento le asocia el correspondiente cambio en la evaluación.
- tactOps: Un Hashmap que asocia un identificador a cada movimiento en el que el oponente cometió un error.
- bestMoves: un Hashmap que a cada posición en formato fen le asocia el correspondiente mejor movimiento según Stockfish.

➤ Métodos de GameAnalysis:

- GameAnalysis(): Un constructor que toma como argumento un objeto de tipo Game e inicializa algunos de los atributos de la clase.
- makeEvaluations(): Este método inicializa Stockfish y recorre cada uno de los movimientos del juego, obteniendo la evaluación de Stockfish para cada posición y el correspondiente mejor movimiento según Stockfish. Este método modifica los atributos moveEvaluations y bestMoves.
- makeDeltaEvaluations(): Este método asigna a cada movimiento el correspondiente cambio en la evaluación y guarda los resultados en el atributo deltaEvaluations.
- makeTactOps(): Este método detecta los errores que el contrincante comete y guarda los resultados en el atributo tactOps.

Clases encargadas del siguiente caso de uso: Ejercicios de ajedrez.

StockFish (Modelo): Esta clase se encarga de conectar con el motor de ajedrez local para pedirle las evaluaciones de las posiciones.

➤ Atributos de StockFish:

- Process: En este atributo se guarda la referencia al proceso de sistema de StockFish.
- Writer: Escritor para solicitar las evaluaciones al proceso de StockFish.
- Reader: Lector de las respuestas de StockFish.

➤ Métodos de StockFish:

- StockFish(): Constructor que inicializa el proceso.

- GetEvaluation(): Método para pedirle la evaluación a StockFish a partir de cierta posición de ajedrez.
- Close(): Cerrar el proceso de StockFish.

```
public class StockFish {
    private Process process;
    private BufferedWriter writer;
    private BufferedReader reader;

    public StockFish() throws IOException {
        // Inicia Stockfish como proceso
        ProcessBuilder processBuilder = new ProcessBuilder("C:\\Users\\kamus\\Docum
        processBuilder.redirectErrorStream(true);
        process = processBuilder.start();

        // Inicializa los flujos de entrada/salida
        writer = new BufferedWriter(new OutputStreamWriter(process.getOutputStream(
        reader = new BufferedReader(new InputStreamReader(process.getInputStream()))

        // Envía el comando "uci" para activar el modo Universal Chess Interface (U
        writer.write("uci\n");
        writer.flush();

        // Espera la confirmación de Stockfish
        String line;
        while ((line = reader.readLine()) != null) {
            if (line.contains("uciok")) break;
        }
    }

    public String[] getEvaluation(String fen) throws IOException {
```

ChessExercises (Modelo): Clase para instanciar los ejercicios de ajedrez:

➤ Atributos de ChessExercises:

- Fen: La posición del ejercicio en formato FEN.
- Level: La dificultad del ejercicio.
- Title: Título del ejercicio.
- MovesForward: Cuántos pasos hay para completar el ejercicio.
- Aim: Objetivo del ejercicio. Ejemplo: dar mate al rey rival.

➤ Métodos de ChessExercises:

- ChessExercises(): Constructor del ejercicio.
- GetFen(): Obtiene el fen del ejercicio.
- GetLevel(): Obtiene el nivel del ejercicio.

- GetMovesForward(): Obtiene la cantidad de pasos para el ejercicio.
- GetAim(): Obtiene el objetivo del ejercicio.

```
public class ChessExercises {
    private String fen;
    private String level;
    private String title;
    private int movesForward;
    private String aim;

    public ChessExercises(String fen, String level, String title, int movesForward, S
        this.fen=fen;
        this.level=level;
        this.title=title;
        this.movesForward=movesForward;
        this.aim=aim;
    }
    //getters
    public String getFen(){
        return this.fen;
    }
    public String getLevel(){
        return this.level;
    }
    public String getTitle(){
        return this.title;
    }
    public int getMovesForward(){
```

OutPuts (Modelo): En esta clase se definen mensajes predeterminados para el usuario. Se definen los mensajes para cuando el usuario hace un buen movimiento o un mal movimiento. Ejemplo: “¡Gran movimiento! sigue así”.

➤ Atributos de OutPuts:

- GoodMoves: Mensajes para buenos movimientos.
- BadMoves: Mensajes para malos movimientos.
- BackMoves: Mensajes para cuando el usuario va hacia atrás en el ejercicio.
- CompletedExerciseMessage: Mensajes para cuando el usuario completa el ejercicio.
- NotBadMoveMessage: Mensajes para movimientos más o menos buenos.
- SlightlyBadMoveMessage: Mensajes para movimientos un poco malos.
- VeryBadMoveMessage: Mensajes para movimientos muy malos.

➤ Métodos para OutPuts:

- `GetGoodMoveMessage()`: Obtiene un mensaje bueno, al azar.
- `GetBadMoveMessage()`: Obtiene un mensaje malo, al azar.
- `GetNotBadMoveMessage()`: Obtiene un mensaje para movimientos más o menos buenos, al azar.
- `GetCompletedExerciseMessage()`: Obtiene un mensaje para cuando el usuario completa el ejercicio, al azar.
- `GetVeryBadMoveMessage()`: Obtiene un mensaje para cuando el usuario hace un movimiento muy malo, al azar.
- `SlightlyBadMoveMessage()`: Obtiene un mensaje para cuando el Movimiento no es tan malo, al azar.
- `GetBackMovesMessage()`: Obtiene un mensaje cuando el usuario va atrás en el ejercicio, al azar.

```
public class OutPuts {
    private static final String[] goodMoves = {
        ";Perfecto! Vas por buen camino.",
        ";Muy bien! La pista tenía razón.",
        ";Excelente elección! Sigue así.",
        ";Acertaste! Esa era la jugada correcta.",
        ";Bien hecho! El camino se aclara.",
        ";Gran jugada! Cada vez más cerca.",
        ";Perfecto! La solución está más cerca.",
        ";Esa es la actitud! Buen movimiento.",
        ";Justo lo que necesitábamos!",
        ";Sigues avanzando con precisión!"
    };

    private static final String[] badMoves = {
        ";Cuidado! Esa no era la mejor opción "
```

ChessExercisesController (Vista-modelo): Esta clase se encarga de interactuar con el proceso de StockFish. Obtiene las evaluaciones y maneja la lógica necesaria para determinar si determinado movimiento de pieza fue bueno o no y de tal manera informar al usuario.

➤ Atributos de ChessExercisesController:

- `StockFish`: referencia a la clase `StockFish()`.
- `ActualStepInExercise`: Guarda el punto del ejercicio en donde se encuentra el usuario.
- `SideOfUser`: Guarda el lado del usuario (blancas o negras).
- `LastEvaluationNumber`: Última evaluación numérica.
- `BestMoveInActualPosition`: Mejor movimiento en la posición actual.

➤ Métodos de ChessExercisesController:

- EvaluateFreePostion(): Permite evaluar una posición configurada libremente por el usuario.
- EvaluatePosition(): Maneja la lógica para determinar si el movimiento de usuario fue bueno o malo.
- GetExercises(): Obtiene los ejercicios predeterminados, clasificados en táctica, finales y ejercicios de mates.
- DoEngineMove(): Hace el movimiento sugerido por la máquina.
- BuildMessage(): Construye el mensaje que se va a mostrar al usuario mediante la clase OutPuts().

```
public class ChessExercisesController {
    private static StockFish stockFish;
    private static int actualStepInExercise=-1;
    private static int totalStepsInExercise=0;
    private static String sideOfUser;
    private static String sideOfMachine;
    private static double errorTolerance=0;
    private static double lastEvaluationNumber=0.5;
    private static String bestMoveInActualPosition;

    static {
        try {
            stockFish = new StockFish();
        } catch (IOException e) {
            e.printStackTrace();
            System.err.println("Error al iniciar StockFish");
        }
    }

    public static void EvaluateFreePosition(String fenBefore,String fenAfter,String fenAfter)
    try{
        if(isEngine && fenBefore==null){
            String[] evaluationAfter=stockFish.getEvaluation(fenAfter);
            NumberFormat format = NumberFormat.getInstance(Locale.FRANCE);
            window.endLoading();
            if(evaluationAfter[2].equals("")!=true){
                int mateStatusAfter=Integer.parseInt(evaluationAfter[2]);
                mateStatusAfter=mateStatusAfter*-1;
                if(mateStatusAfter==0){
```

Clases encargadas del siguiente caso de uso: Recursos de ajedrez.

ChessResource (Modelo): Clase para instanciar un recurso de ajedrez, ejemplo de recursos: videos de youtube, libros y artículos.

➤ Atributos de ChessResource:

- Title: título del recurso.

- Type: Determina el tipo del recurso (Video, artículo, libro, etc).
- Phase: Determina la fase del juego de la cual se trata el recurso; apertura, medio juego o final.
- Url: Determina la dirección al recurso.

➤ Métodos de ChessResource:

- ChessResource(): Constructor del recurso.
- GetTitle(): Obtiene el título del recurso.
- GetType(): Obtiene el tipo de recurso.
- GetPhase(): Obtiene de qué se trata el recurso.
- GetUrl(): Obtiene la url del recurso.

```
public class ChessResource {
    private String title;
    private String type; // Libro o Video
    private String phase; // Apertura, Medio Juego, Final
    private String url;

    public ChessResource(String title, String type, String phase, String url) {
        this.title = title;
        this.type = type;
        this.phase = phase;
        this.url = url;
    }

    public String getTitle() {
        return title;
    }

    public String getType() {
        return type;
    }

    public String getPhase() {
```

ChessResourceModel (Modelo): En esta clase se instancian los recursos que se van a presentar al usuario.

➤ Atributos de ChessResourceModel:

- Resources: Lista con los recursos a devolver.

➤ Métodos de ChessResourceModel:

- GetResourcesByPhase(): Obtiene los recursos filtrando por la fase del juego.
- GetAllResources(): Obtiene todos los recursos.

```
public class ChessResourceModel {
    private List<ChessResource> resources;

    public ChessResourceModel() {
        resources = new ArrayList<>();
        loadResources();
    }

    private void loadResources() {
        resources.add(new ChessResource("Los mejores LIBROS de APERTURAS de AJEDREZ", "Libro", "Apertura"));
        resources.add(new ChessResource("El ajedrez de torneo", "Libro", "Apertura"));
        resources.add(new ChessResource("El arte del medio juego", "Libro", "Medio juego"));
        resources.add(new ChessResource("Los 10 MEJORES libros de FINALES de AJEDREZ", "Libro", "Apertura"));
        resources.add(new ChessResource("5 MEJORES APERTURAS para aficionados (Intermedio)", "Libro", "Apertura"));
        resources.add(new ChessResource("5 TRAMPAS de apertura 'buenisimas' en ajedrez", "Libro", "Apertura"));
        resources.add(new ChessResource("Las mejores APERTURAS de AJEDREZ", "Video", "Apertura"));
        resources.add(new ChessResource("5 MEJORES aperturas para el 2024 (BLANCAS)", "Video", "Apertura"));
        resources.add(new ChessResource("Las 5 aperturas de ajedrez más agresivas", "Video", "Apertura"));
        resources.add(new ChessResource("Aperturas de ajedrez", "Video", "Apertura"));
        resources.add(new ChessResource("La mejor apertura blanca Sub-1600 Elo", "Video", "Apertura"));
        resources.add(new ChessResource("5 MEJORES aperturas para el 2024 (NEGRAS)", "Video", "Apertura"));
        resources.add(new ChessResource("5 trucos de apertura para ganar al ajedrez", "Video", "Apertura"));
        resources.add(new ChessResource("La MEJOR apertura contra 1.e4 - ¡Una trampa!", "Video", "Apertura"));
        resources.add(new ChessResource("Domina las aperturas de ajedrez y sé un maestro", "Video", "Apertura"));
        resources.add(new ChessResource("REPERTORIO COMPLETO de APERTURAS de AJEDREZ", "Video", "Apertura"));
        resources.add(new ChessResource("La MEJOR TRAMPA de apertura contra 1.e4", "Video", "Apertura"));
        resources.add(new ChessResource("Apertura UNIVERSAL 1.b3 para las blancas", "Video", "Apertura"));
        resources.add(new ChessResource("Las mejores aperturas de ajedrez y cómo elegir la tuya", "Video", "Apertura"));
        resources.add(new ChessResource("Aperturas de ajedrez | Chesscul", "Artículo", "Apertura"));
        resources.add(new ChessResource("5 MEJORES aperturas para el 2024 (BLANCAS)", "Video", "Apertura"));
        resources.add(new ChessResource("Las 5 aperturas de ajedrez más agresivas", "Video", "Apertura"));
        resources.add(new ChessResource("Aperturas de ajedrez", "Video", "Apertura"));
    }
}
```

ChessResourcesController (Vista-modelo): Esta clase se comunica con la vista para desplegar los recursos.

➤ Atributos de ChessResourcesController:

- View: instancia de la vista de recursos.
- ChessResourceModel: instancia de la clase ChessResourceModel.

➤ Métodos de chessResourcesController:

- ChessResourcesController(): Constructor de la clase.
- LoadResources(): Carga los recursos de ajedrez a la vista.

```
public class ChessResourcesController {
    private ChessResourcesView view;
    private ChessResourceModel model;

    public ChessResourcesController(ChessResourcesView view, ChessResourceModel model) {
        this.view = view;
        this.model = model;
        loadResources();
    }

    private void loadResources() {
        List<ChessResource> openingResources = model.getResourcesByPhase("Apertura");
        List<ChessResource> middlegameResources = model.getResourcesByPhase("Medio Juego");
        List<ChessResource> endgameResources = model.getResourcesByPhase("Final");

        view.updateSection("Apertura", openingResources);
        view.updateSection("Medio Juego", middlegameResources);
        view.updateSection("Final", endgameResources);
    }
}
```

ChessResourcesView (Vista): Esta es la vista de los recursos de ajedrez.

➤ Atributos de la clase ChessResourcesView:

- OpeningPanel: Sección de recursos de apertura de ajedrez.
- MiddlegamePanel: Sección de recursos de mitad de juego de ajedrez.
- EndgamePanel: Sección de recursos de finales de ajedrez.

➤ Métodos de la clase ChessResourcesView:

- ChessResourcesView(): Constructor de la vista.
- CreateSection(): Crea la sección para cada fase del juego.
- UpdateSection(): Actualiza la vista de cada sección.


```

public class ChessResourcesView extends JFrame {

    private JPanel openingPanel;
    private JPanel middlegamePanel;
    private JPanel endgamePanel;

    public ChessResourcesView() {
        setTitle("Recursos de Ajedrez");
        setSize(900, 700);
        getContentPane().setBackground(new Color(213, 249, 222));
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Menú
        JMenuBar menuBar = new JMenuBar();
        JMenu menu = new JMenu("Navegación");
        menu.setForeground(new Color(10, 10, 10));
        JMenuItem exercisesItem = new JMenuItem("Ejercicios");
        exercisesItem.addActionListener(e -> {
            dispose();
            SwingUtilities.invokeLater(() -> new ExercisesView()); // Asegúrate de t
        });

        JMenuItem resourcesItem = new JMenuItem("Recursos");
        resourcesItem.addActionListener(e -> {

        });


        menu.add(exercisesItem);
        menu.add(resourcesItem);
        menuBar.add(menu);
    }
}

```

Manual de Usuario

Autenticación del usuario: El usuario ve como primer pantalla la vista de autenticación, donde deberá hacer los siguiente:

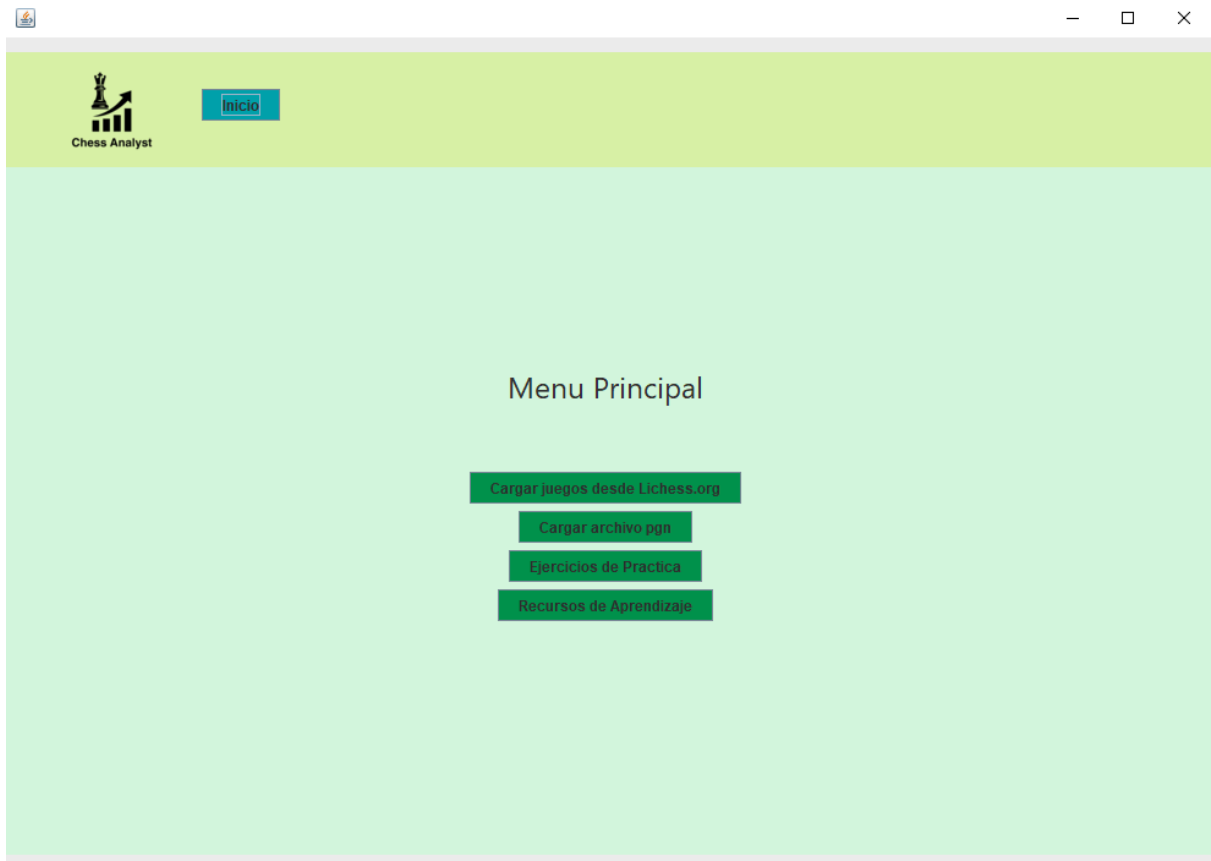
- Registrarse si no tiene cuenta.
- Iniciar sesión si ya tiene cuenta.



The screenshot shows a web browser window titled "Authentication". The background is light green. At the top, it says "Bienvenido, por favor regístrate o inicia sesión si ya tienes una cuenta". Below this, the text "Iniciando sesión" is centered. There are two input fields: "Username:" with the value "juanRo" and "Password:" with masked characters ".....". Below the fields are two buttons: "Log In" (green) and "Register" (grey).

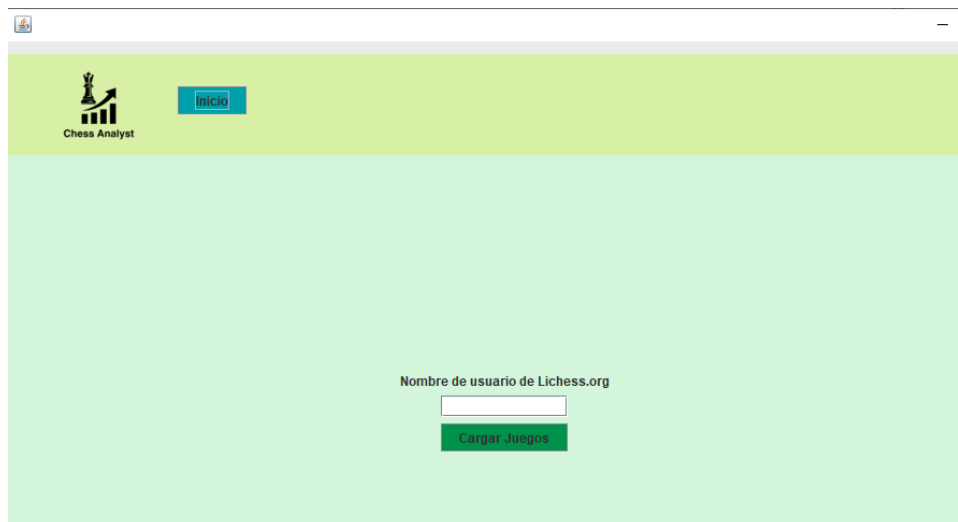
Menú principal

Después de la autenticación del usuario, el menú principal se despliega y el usuario tiene la posibilidad de seleccionar desde qué fuente cargará los juegos, si desea realizar ejercicios de práctica o si desea ver recursos para el aprendizaje:



Cargar Juegos desde Lichess.org

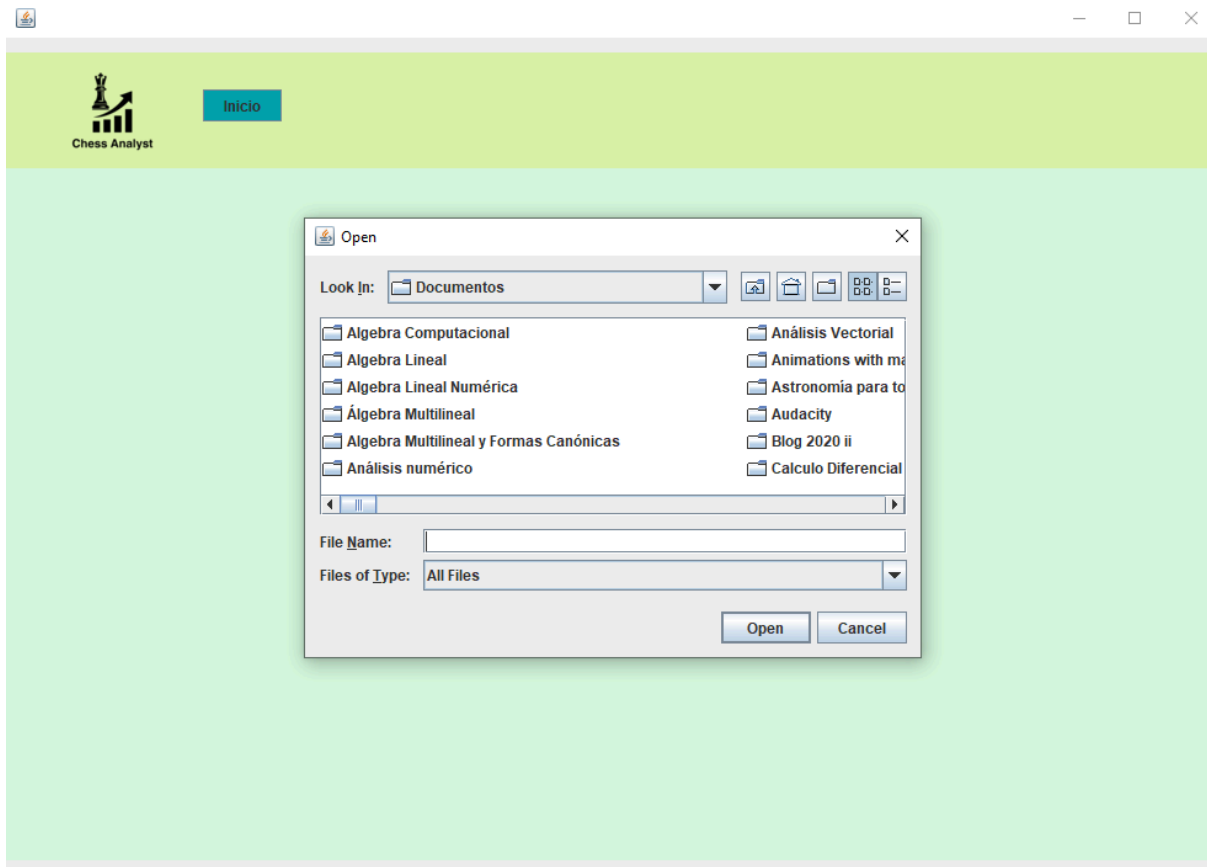
Si el usuario tiene una cuenta en lichess.org puede cargar los juegos directamente desde la plataforma, seleccionando la opción Cargar juegos desde Lichess.org:



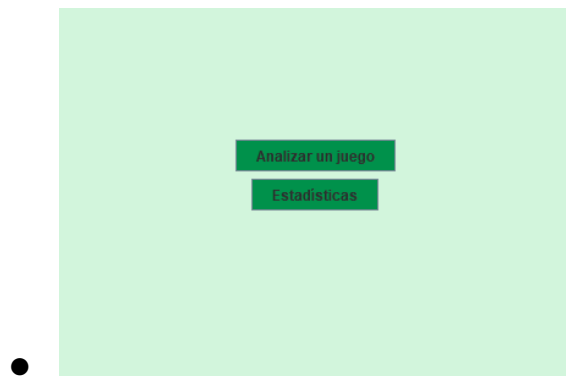
Aquí el usuario ingresa su nombre de usuario en lichess.org y oprime el botón Cargar Juegos. Alternativamente, el usuario puede cargar los juegos desde un archivo .png que tenga en su equipo seleccionando la opción Cargar Archivo png en el menú principal.

Cargar Juegos desde un archivo .png

En esta ventana el usuario puede navegar en el sistema de archivos de su computador y seleccionar un archivo .png en el que estén los juegos que desea analizar

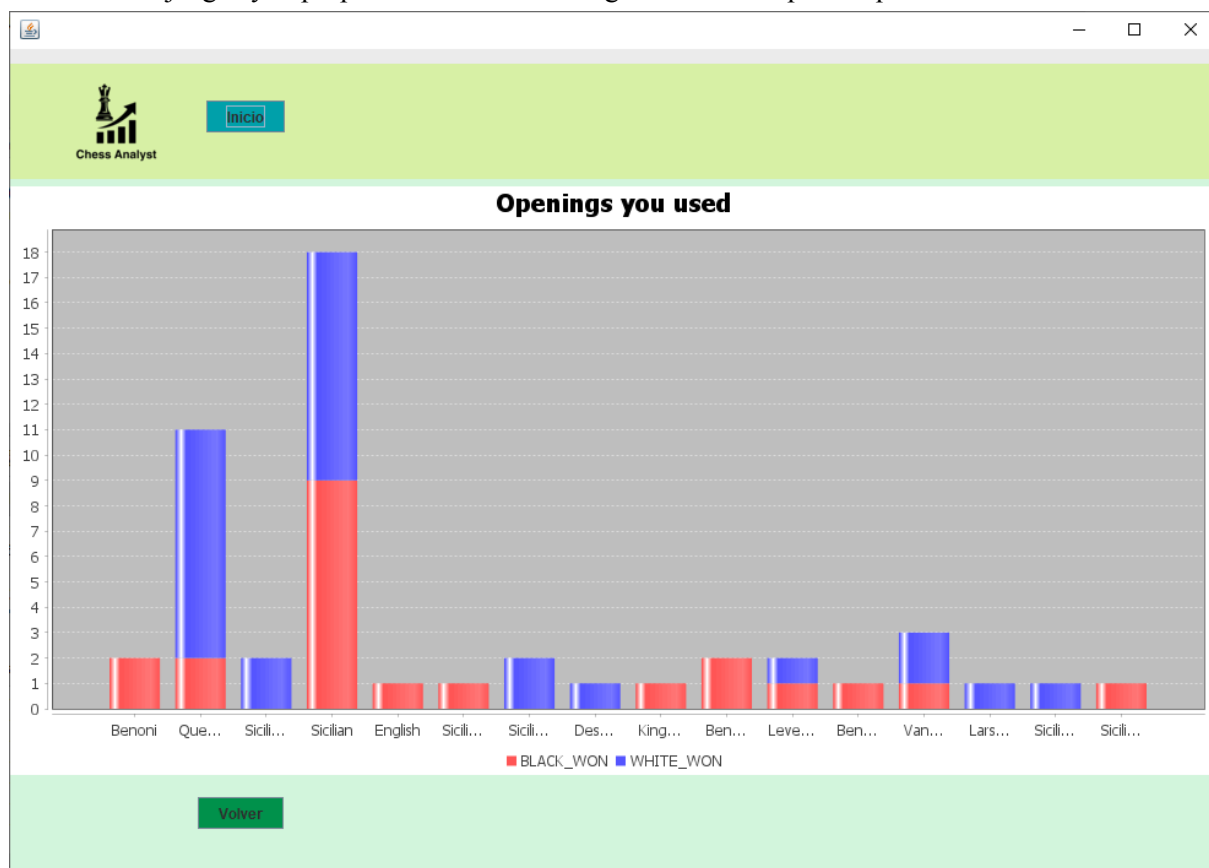


Una vez cargados los juegos se despliega una ventana con las opciones del análisis.



Estadísticas

La opción Estadísticas permite al usuario ver una gráfica con las principales aperturas que aparecen dentro de sus juegos y la proporción de victorias según el color de piezas que el usuario tiene:

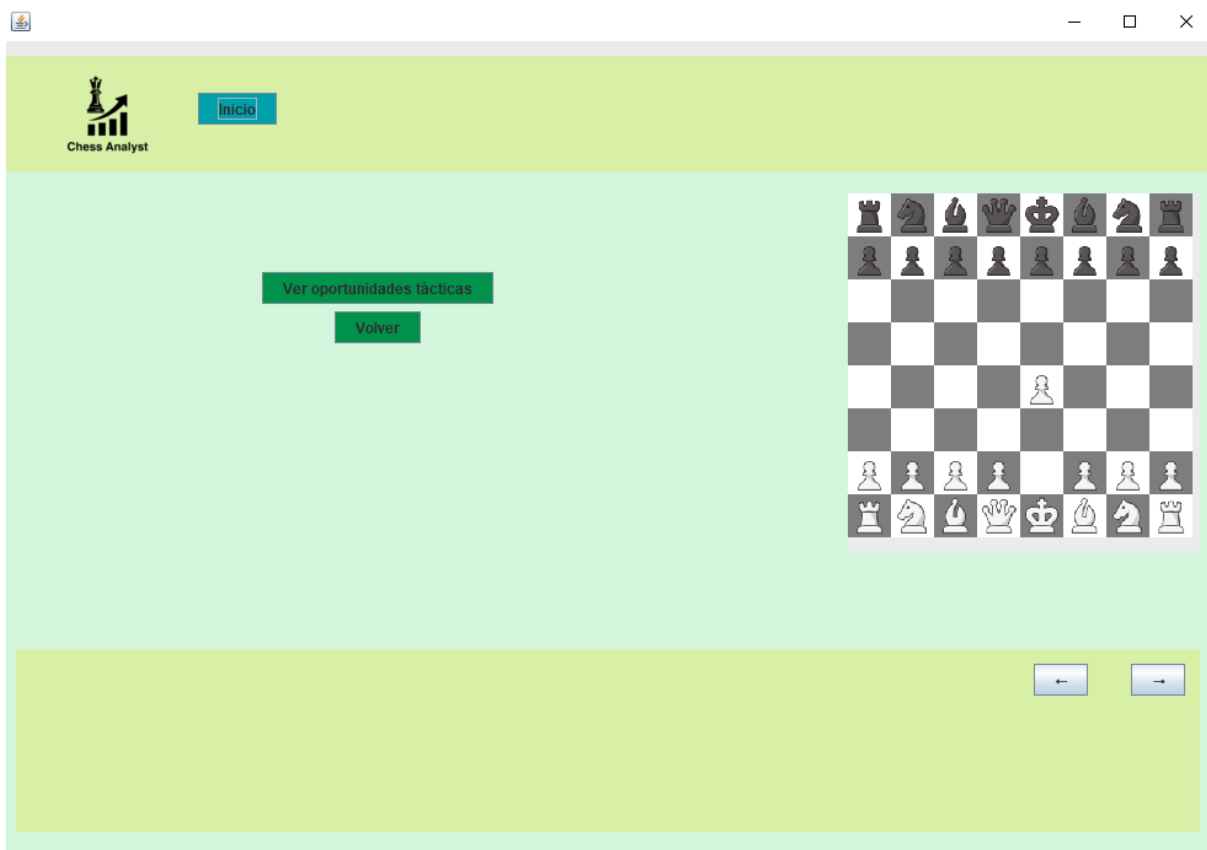


Analizar un juego

Si el usuario escoge la opción analizar un juego se despliega una ventana con una lista de los juegos contenidos en el archivo .png o recuperados desde la plataforma lichess.org:

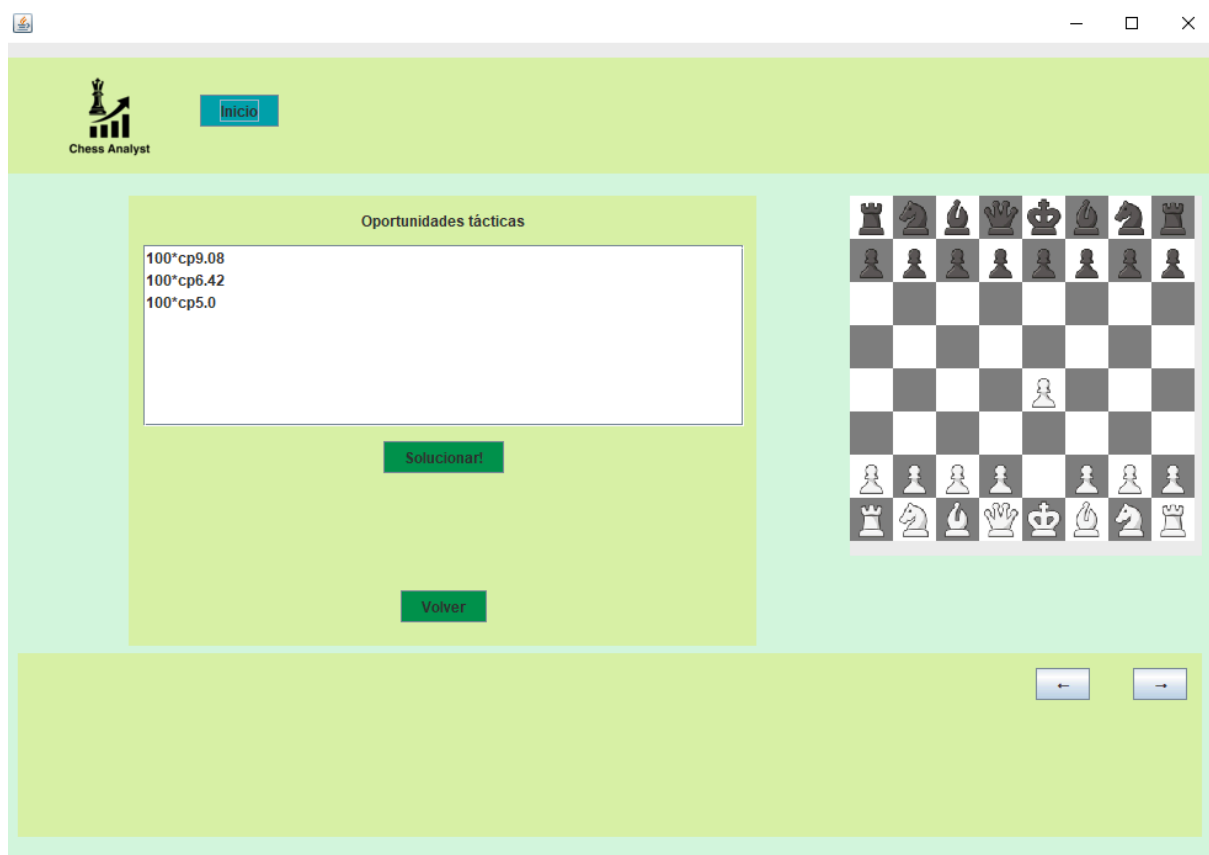


El usuario selecciona un juego dentro de la lista y da click en el botón Analizar! A continuación se muestra un tablero de ajedrez y los botones para avanzar o retroceder en los movimientos del juego seleccionado, los cuales se muestran en el tablero.

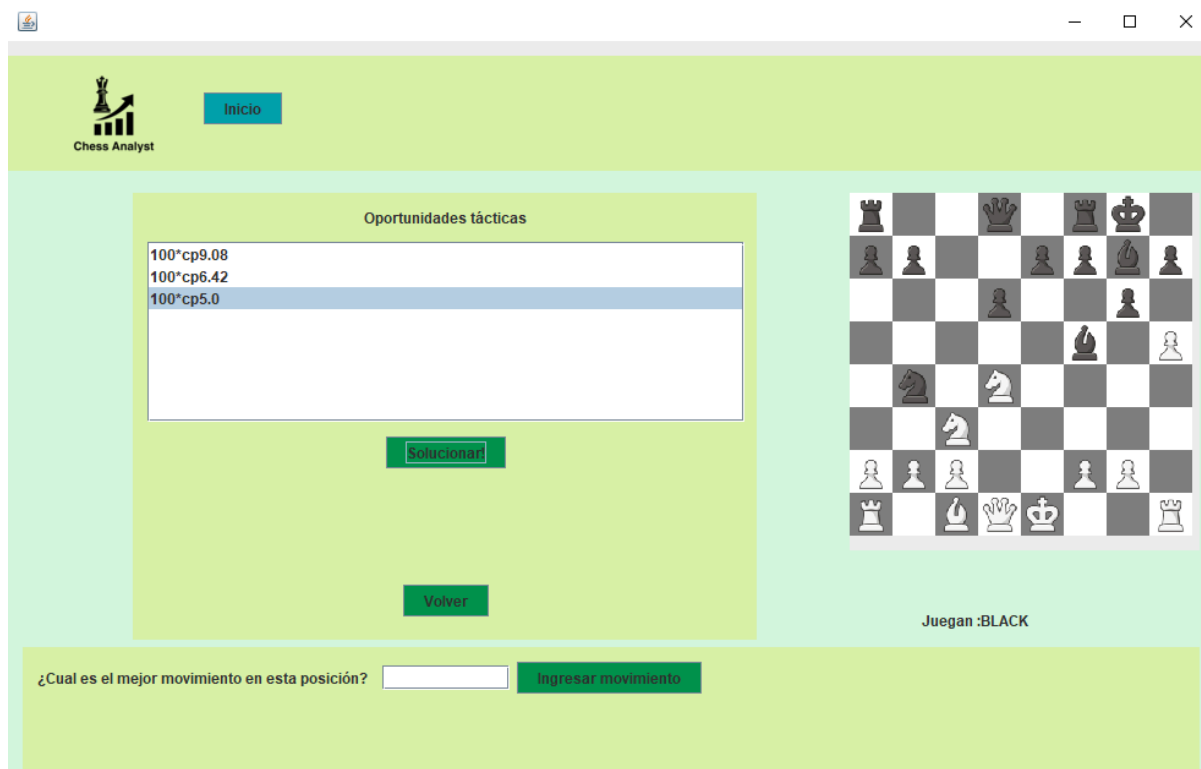


Ver oportunidades tácticas

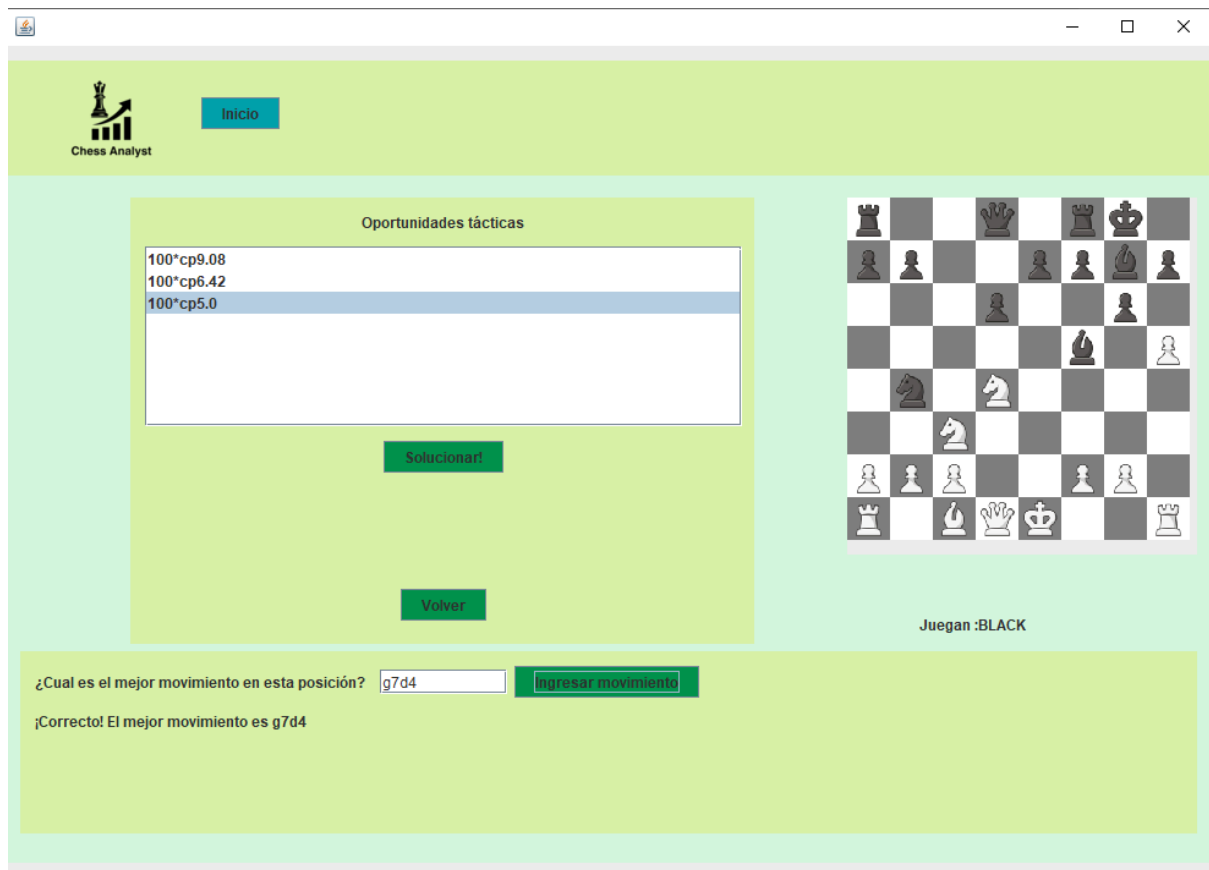
Cuando el usuario da click en el botón ver oportunidades tácticas se despliega una lista con las posiciones en las que el usuario tiene la oportunidad de conseguir una ventaja considerable debido a que su oponente cometió un error en la jugada inmediatamente anterior a dicha posición. Cada elemento de la lista muestra el cambio en la evaluación del juego (en centí peones abreviado cp) debido al error del contrincante, lo cual da una medida de la ventaja que el usuario ahora puede explotar para ganar el juego. Entre mayor sea el número de centí peones mayor es la ventaja que el usuario ha ganado. El propósito de esta dinámica es permitirle al usuario practicar la detección de oportunidades tácticas basándose en sus propios juegos.



Cuando el usuario selecciona un elemento de la lista y da click en Solucionar! se muestra la posición seleccionada en el tablero y un campo para que el usuario ingrese el movimiento que considera es el mejor para aprovechar la oportunidad táctica:

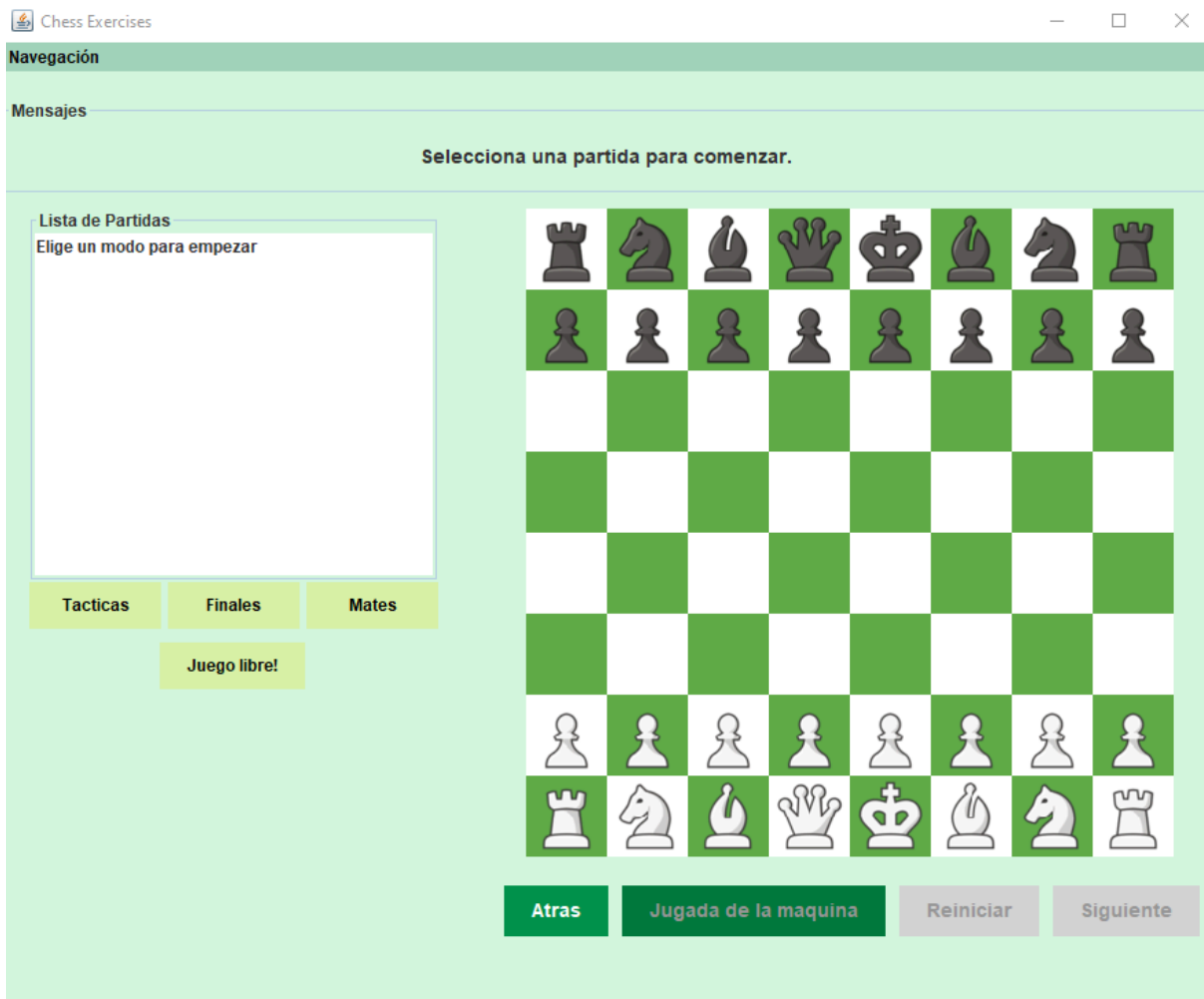


Si el usuario ingresa el mejor movimiento en la posición, el cuál se obtuvo durante el análisis del juego efectuado con Stockfish, aparece un mensaje indicando que dicho movimiento es acertado, de lo contrario aparece un mensaje señalando que la selección del usuario es incorrecta y mostrando el verdadero mejor movimiento:

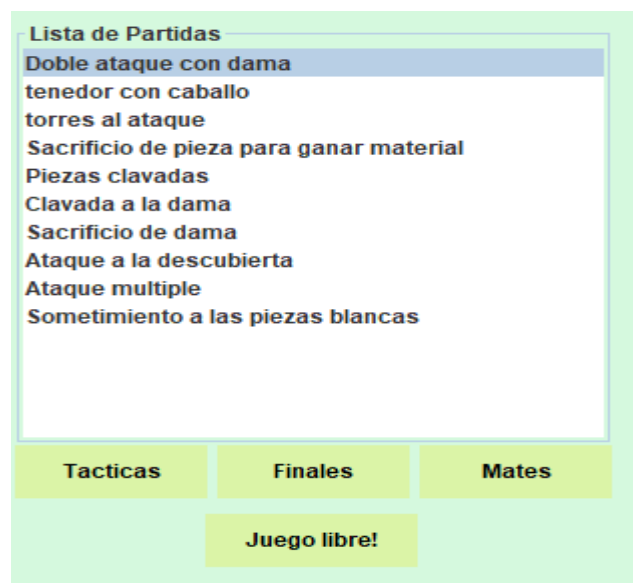


Ejercicios de ajedrez

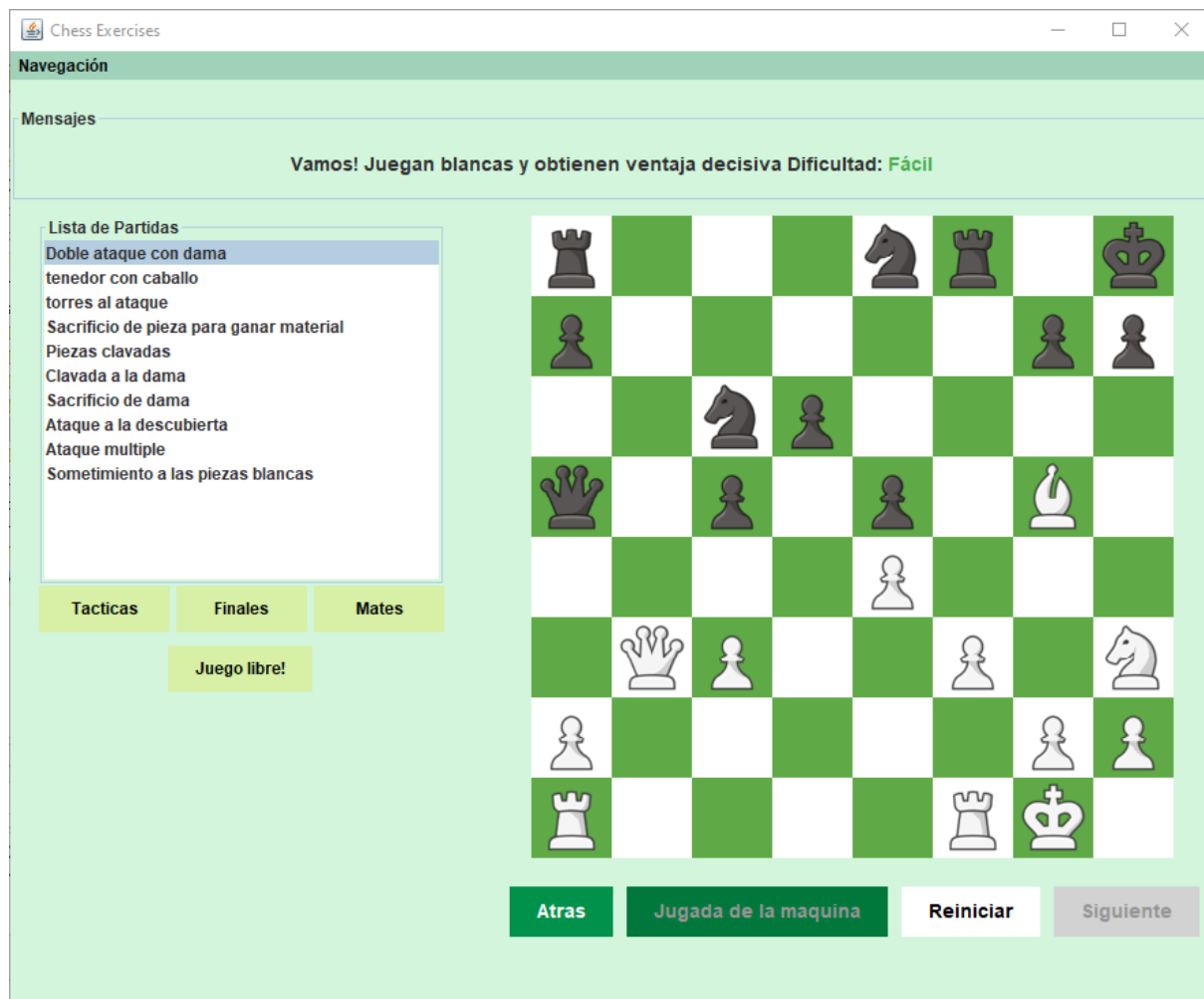
En el menú principal, al dar click en la opción Ejercicios de Práctica, se despliega la pantalla de ejercicios, donde el usuario podrá interactuar con cada ejercicio propuesto y analizar una posición configurada libremente por él.



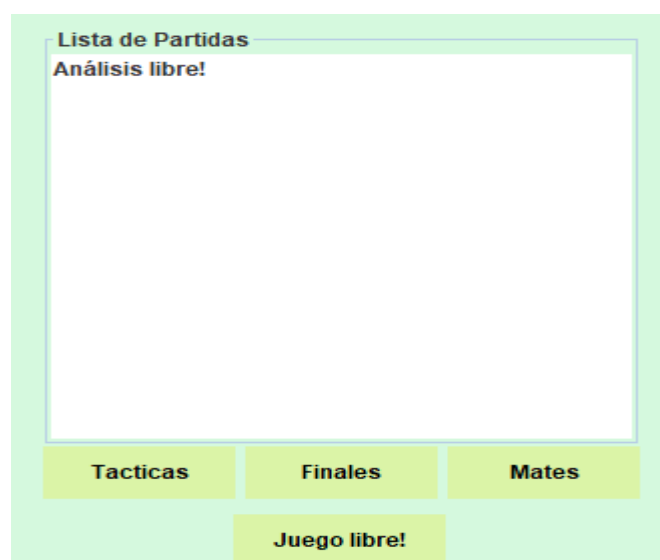
Se muestran los ejercicios específicos del modo que elige el usuario:



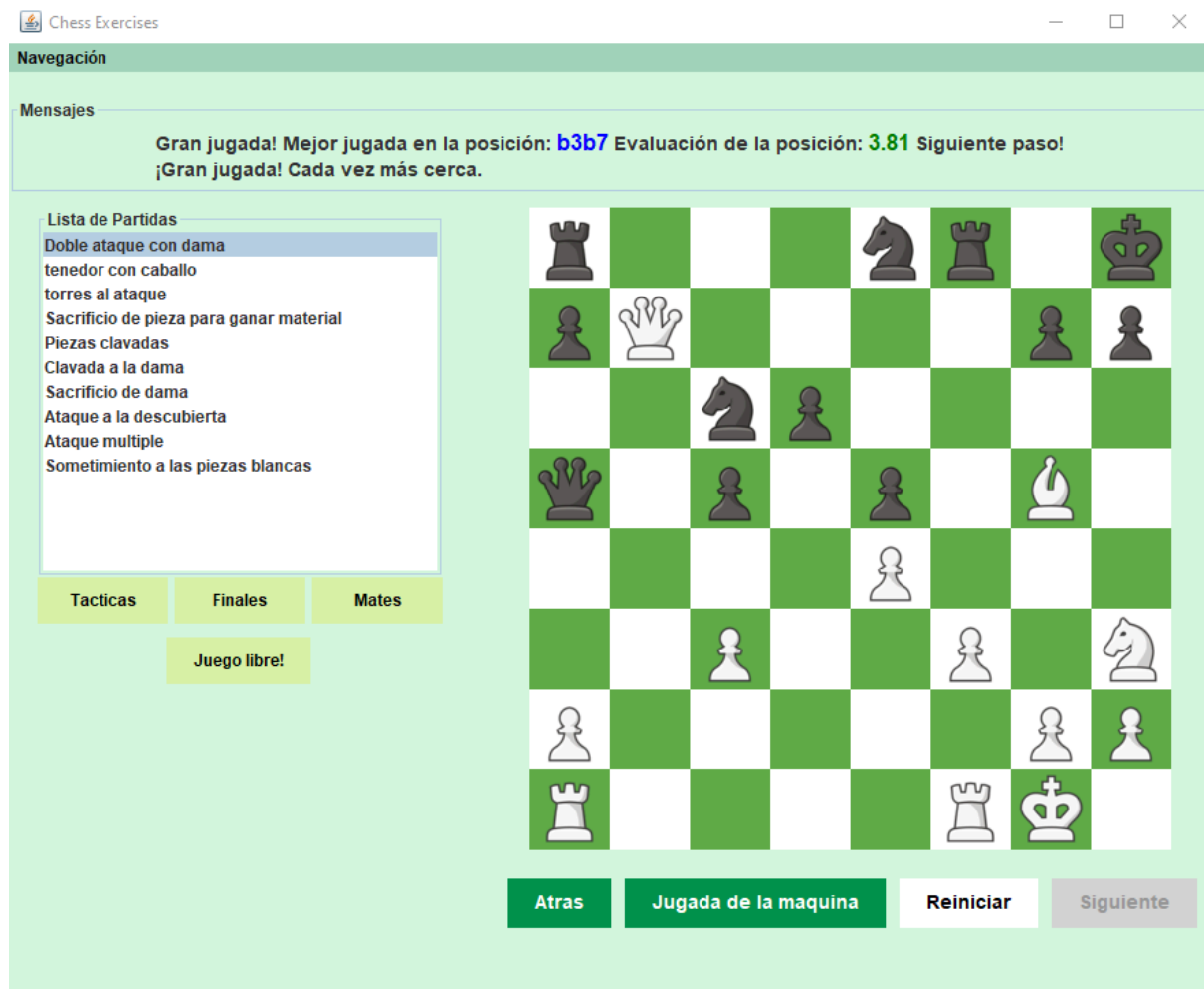
El usuario selecciona un ejercicio en específico y se muestra en el tablero:



nota: como adición al caso de uso, se coloca la opción de juego libre! para que el usuario pueda crear una posición cualquiera y recibir retroalimentación.



El usuario hace la jugada que considera correcta en la posición arrastrando la pieza desde el tablero y recibe retroalimentación de la mejor jugada y de la evaluación numérica de la posición:



El usuario navega a la pantalla de resources mediante el menú de la parte superior de la pantalla, dando clic en navegación y eligiendo la vista:



el usuario ve la vista de recursos de ajedrez en cada de navegar a esa vista:

Recursos de Ajedrez

Navegación

Recursos de Ajedrez: Apertura, Medio Juego y Final

Apertura

Los mejores LIBROS de APERTURAS de AJEDREZ - <https://www.youtube.com/watch?v=-rQD-Mx6u1o>
El ajedrez de torneo - <https://thezugzwangblog.com/los-5-mejores-libros-de-ajedrez-para-jugadores-de-club/>
5 MEJORES APERTURAS para aficionados (Intermedio) - <https://www.youtube.com/watch?v=SvyNX-XzK1I>
5 TRAMPAS de apertura 'buenisimas' en ajedrez BALA - https://www.youtube.com/watch?v=T6a96_ZaTzY
Las mejores APERTURAS de AJEDREZ - https://www.youtube.com/playlist?list=PLMWPPmJLOBgaq8uajHHqa3Syh_BPgZvjh
5 MEJORES aperturas para el 2024 (BLANCAS) - <https://www.youtube.com/watch?v=rDblLvBYrPc>
Las 5 aperturas de ajedrez más agresivas y sorprendentes - <https://www.chess.com/es/article/view/las-5-aperturas-de-ajedrez-mas-agresivas-y-sorp>
Aperturas de ajedrez - <https://www.youtube.com/playlist?list=PLH6dDsirf5Z2n1ArynBRIRScnzz2suE4n>

Medio Juego

El arte del medio juego - https://www.reddit.com/r/chess/comments/1ec1e20/what_booksvideos_should_i_readwatch_to_improve_my/
Estrategia en ajedrez: Cómo planificar en el medio juego - <https://chesscul.com/estrategia-ajedrez/>
5 tácticas fundamentales en el medio juego #2 - <https://www.chess.com/es/blog/Adriwif/5-tacticas-fundamentales-en-el-medio-juego-2>
Medio juego (ajedrez) - https://es.wikipedia.org/wiki/Medio_juego_%28ajedrez%29
Elaboración de planes en el medio juego - <https://www.youtube.com/watch?v=f1rlqk1efjs>
Estrategias de Medio Juego que Todo Jugador de Ajedrez Debe Conocer - <https://clubajedrezlima.com/2024/05/25/estrategias-de-medio-juego-que-tod>
Una regla vital para el medio juego - <https://www.chess-teacher.es/una-regla-vital-medio-juego/>
El Ajedrez en el Medio Juego - <https://www.astridvasquezci.com/el-ajedrez-en-el-medio-juego/>

Final

Los 10 MEJORES libros de FINALES de AJEDREZ - https://www.youtube.com/watch?v=V_li2OrVUSU
5 recursos mágicos en los finales de ajedrez - <https://www.chess.com/es/article/view/5-recursos-magicos-en-los-finales-de-ajedrez>
5 recursos ESENCIALES en los FINALES de ajedrez - <https://www.youtube.com/watch?v=gRbfsSC6FXU>
Finales de ajedrez: conceptos clave para dominar la fase final - <https://www.superprof.cr/blog/finales-ajedrez/>
3 FINALES MILAGROSOS ¡Aprende estos recursos! - https://www.youtube.com/watch?v=Egs_jP_koU
7 elementos fundamentales en los finales de ajedrez - <https://capakhine.es/index.php/blog/186-7-elementos-fundamentales-en-los-finales-de-ajedrez>
Juegos con Finales de Ajedrez | Practica y Mejora tu Técnica - <https://www.ajedrezeureka.com/category/juegos-con-finales-de-ajedrez/>
Mejora Tus Finales de Ajedrez Con Este Simple Secreto! - <https://www.youtube.com/watch?v=2CPyL0OW9dg>
Finales de ajedrez: conceptos clave para dominar la fase final - <https://www.superprof.cr/blog/finales-ajedrez/>