I have structured this assessable practice very similarly to how I structured the XML one. From the app class, I start the application and display the menu. I have a class called ChessPlayer to create objects of that type and query their attributes. Another class, ChessTournament, encompasses all the functions related to the collection, manipulation, storage, and deletion of player data. In this case, there are two other classes: one called SQLConnection, in which I have created all the necessary methods to handle the connection with the SQLite database, and another called DBMongoDB, in which I have created, just like with SQL, all the methods related to the connection and manipulation of the MongoDB database. At first, I thought about creating a global variable for the connection and the statement to have it available for all methods and avoid opening and closing the connection all the time. However, I realized that it could be a security issue. So, for each method in which I need a database connection, both for MongoDB and SQL, I open the connection, perform the necessary tasks, and then close the connection again. This makes it a bit more challenging to divide the code into different methods to avoid repeating lines, but I believe that, in the end, it improves security, and the overall result is better.

The main difficulty I have encountered has been adapting to a new type of project and its new protocols: finding the correct dependencies for the versions of SQLite and MongoDB that I have installed on my computer and adapting some functions that, although they appeared in the class notes, were obsolete and had to be done differently. Another difficulty encountered, as in the first assessable practice, has been understanding the many steps required to connect to the database, although it is fair to say that this time it has been easier and less tedious. Understanding the entire process and adapting and customizing everything the way I wanted it to be has been a challenge. For example, one of the things I have changed compared to the class notes is that instead of avoiding the automatic ID that MongoDB assigns, I have converted the player's ID into MongoDB's _id, so now I don't have to exclude it when reading the document. Another difficulty I encountered was the fact that, as it appeared in the notes, when I created a new list of players and dumped it into the MongoDB database, the new players were added to the end of the existing ones. Therefore, I had to implement a function to delete all the documents within the "players" collection and then insert the ones extracted from SQLite so that both databases would match in the end.

An extension that comes to mind for the application is that when I delete players from SQLite, they should also be removed from MongoDB, even though it wasn't required in the practice. But the most important extension I can think of is generating an XML document from the data in one of the databases, which, with an XSL template, allows for the creation of an HTML for presenting the player data in a more user-friendly way.