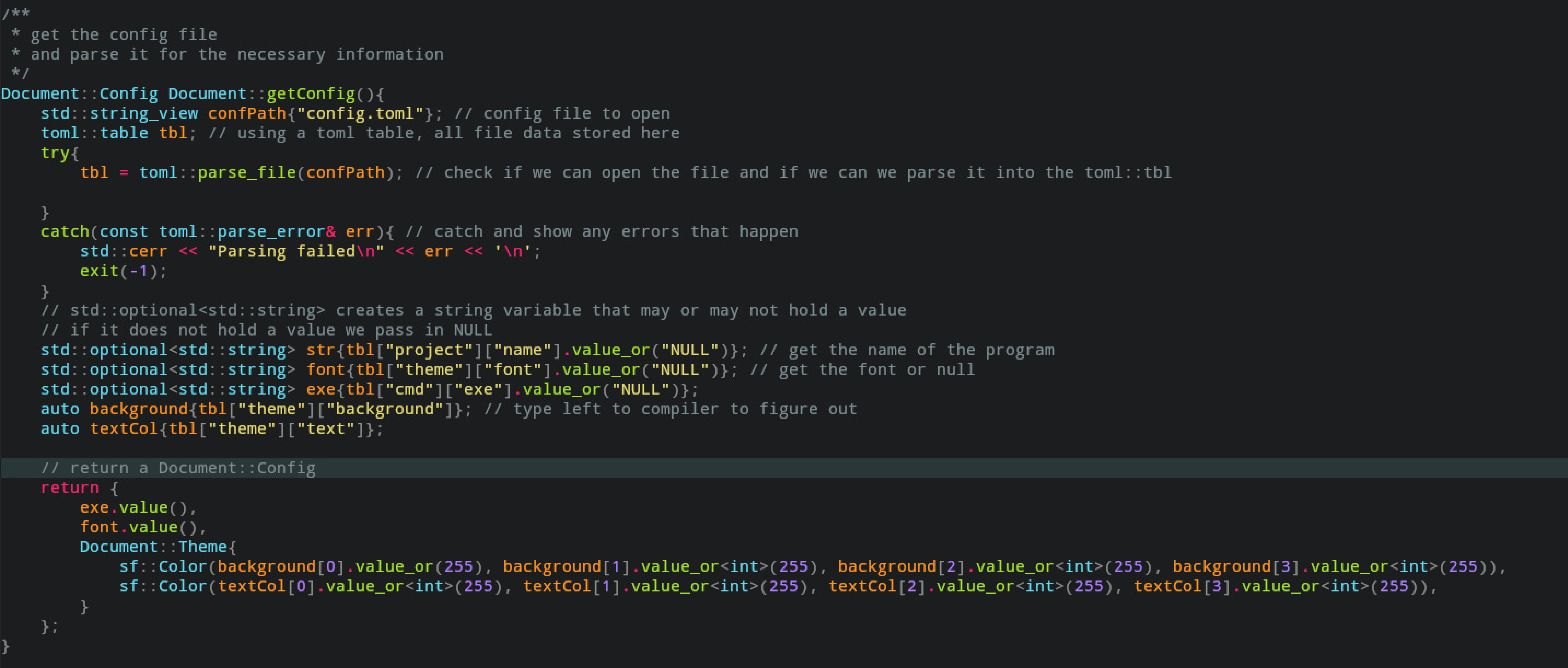

```cpp
/**
 * get the config file
 * and parse it for the necessary information
 */
Document::Config Document::getConfig(){
    std::string_view confPath{"config.toml"}; // config file to open
    toml::table tbl; // using a toml table, all file data stored here
    try{
        tbl = toml::parse_file(confPath); // check if we can open the file and if we can we parse it into the toml::tbl

    }
    catch(const toml::parse_error& err){ // catch and show any errors that happen
        std::cerr << "Parsing failed\n" << err << '\n';
        exit(-1);
    }
    // std::optional<std::string> creates a string variable that may or may not hold a value
    // if it does not hold a value we pass in NULL
    std::optional<std::string> str{tbl["project"]["name"].value_or("NULL")}; // get the name of the program
    std::optional<std::string> font{tbl["theme"]["font"].value_or("NULL")}; // get the font or null
    std::optional<std::string> exe{tbl["cmd"]["exe"].value_or("NULL")};
    auto background{tbl["theme"]["background"]}; // type left to compiler to figure out
    auto textCol{tbl["theme"]["text"]};

    // return a Document::Config
    return {
        exe.value(),
        font.value(),
        Document::Theme{
            sf::Color(background[0].value_or(255), background[1].value_or<int>(255), background[2].value_or<int>(255), background[3].value_or<int>(255)),
            sf::Color(textCol[0].value_or<int>(255), textCol[1].value_or<int>(255), textCol[2].value_or<int>(255), textCol[3].value_or<int>(255)),
        }
    };
}
```