Malphas

Second Revision

by Rastislav Patak
Piotr Wyrwas

May 2025

Architecture Changes

This revision of *Malphas* has subtely altered the architecture of the project and its arrangement of submodules. The most significant change is the introduction of a **Proxt Server**. The reason for this new module was the introduction of authentication via **OAuth**, and was further motivated by the lack of motivation of the developers to implement said authentication facilities in `C++`, which was and still is the preferred backend language of choice. Thus, a strategy was devised to outsource this daunting responsibility onto a lightweight proxy. The proxy itself is written in `Kotlin` and is powered by `Ktor`, a minimal and lightweight web framework designed for implementing microservices.

The proxy implements two additional routes, namely `/login` and `/callback`, which are used for **OAuth**: The `/login` route redirects the user to Google's authentication frontend[1] which, upon completion, redirects the user back to `/callback`. This is where the access token is acquired and the user is sent an authentication cookie containing the aforementioned token, their user ID, as well as an **OAuth** "state". The user is subsequently redirected back to the *Malphas* frontend. Besides the two aforementioned routes, the proxy passess all requests onto the original backend at a configurable `URL`, where they are dealt with accordingly. However, the proxy will only forward requests containing the authentication cookie with a valid access token, unless the route is explicitly declared public in the proxy config.

Since the entirety of the authentication process has been delegated away to another service, parts of the frontend as well as the original backend which were previously responsible for this process could be removed entirely. As such, the authentication endpoints were removed from the original backend. A database migration was performed to remove any tables and remaining columns that previously dealt with authentication. Since Google relies on string values for identifying their users, the type of the `ID` field was also changed to `TEXT`.

[1] Google is the OAuth provider we decided to go with for this project.

Known Bugs

> "*It deletes more than it should, but in the context of its functionality, that's better than deleting too little.*"
>
> – Rasti, 2025, On the exemplary stability of Malphas

The above quote demonstates just one of the many known bugs in the first version of Malphas. And while countless issues and curious behaviours still plague the entire frontend, tha particular bug mentioned in the quote has successfully been tracked down and resolved. Although the exact reasons for why this fix works are between `JavaScript` and God, what seems to have caused was re-building the `AST` inside of the callback of `traverseAllAsts(...)`, i.e.

```
traverseAllAsts(this.ast, (node) => {
        ...
        if ('location' in node) {
                if (this.rayCast(this.snappedMousePosition, node.element.geometry()
                                as unknown as [number, number][], node.location)) {
                        deleteCircuit(deleteNode, () => {
                                this.rebuildAst()
                        })
                }
        }
})
```

It is worth mentioning that the way `deleteCircuit` is implemented, is that the lambda callback is only triggered after an asynchronous request completes, which may explain the undefined behaviour.

The Login Experience

The following visual report demonstates the seamless login experience that our users can expect from *Malphas*:

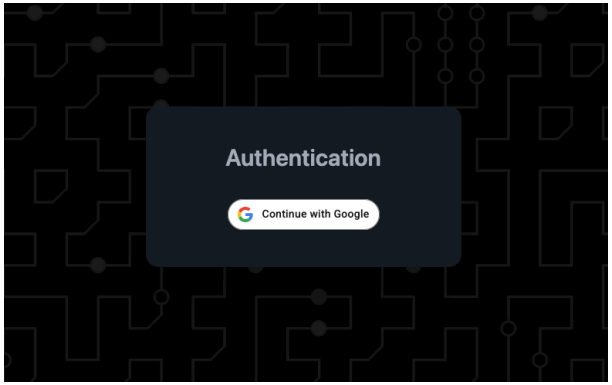

Figure 1: Malphas login screen

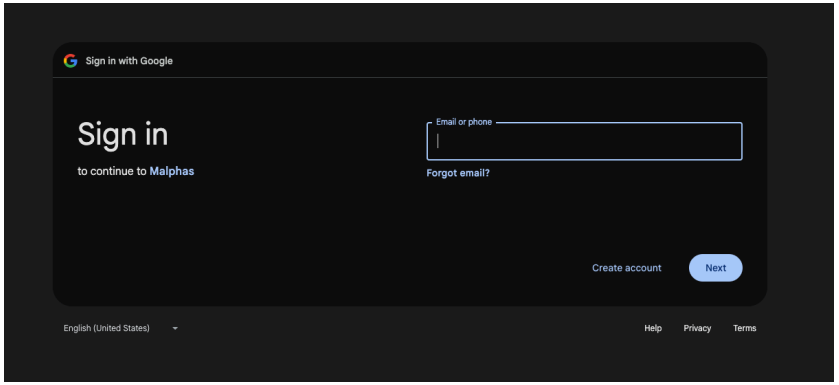

Figure 2: Google's Initial Auth Page

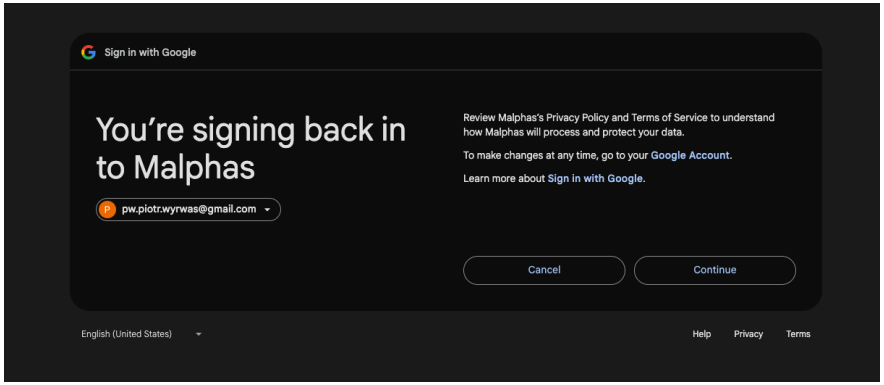

Figure 3: Post-authentication Followup Screen

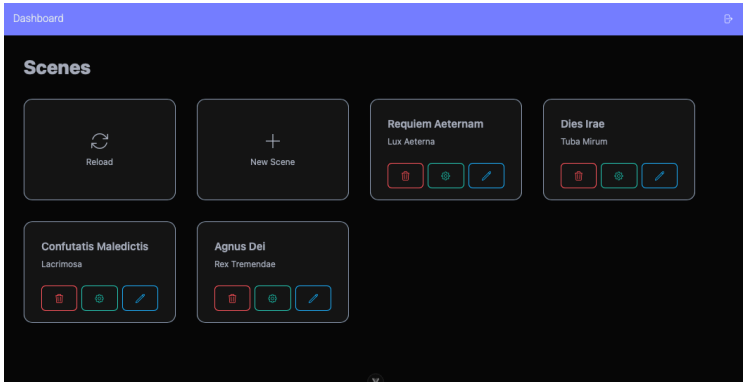

Figure 4: Redirect to Malphas Dashboard