

EVPlanner.rs

*Dennis Orlando, Alessio Zeni,
Fabio Giovanazzi, Filippo De Grandi*



Our team (grazie Marco per le foto <3)



FABIO GIOVANAZZI



FILIPPO DE GRANDI



ALESSIO ZENI



DENNIS ORLANDO

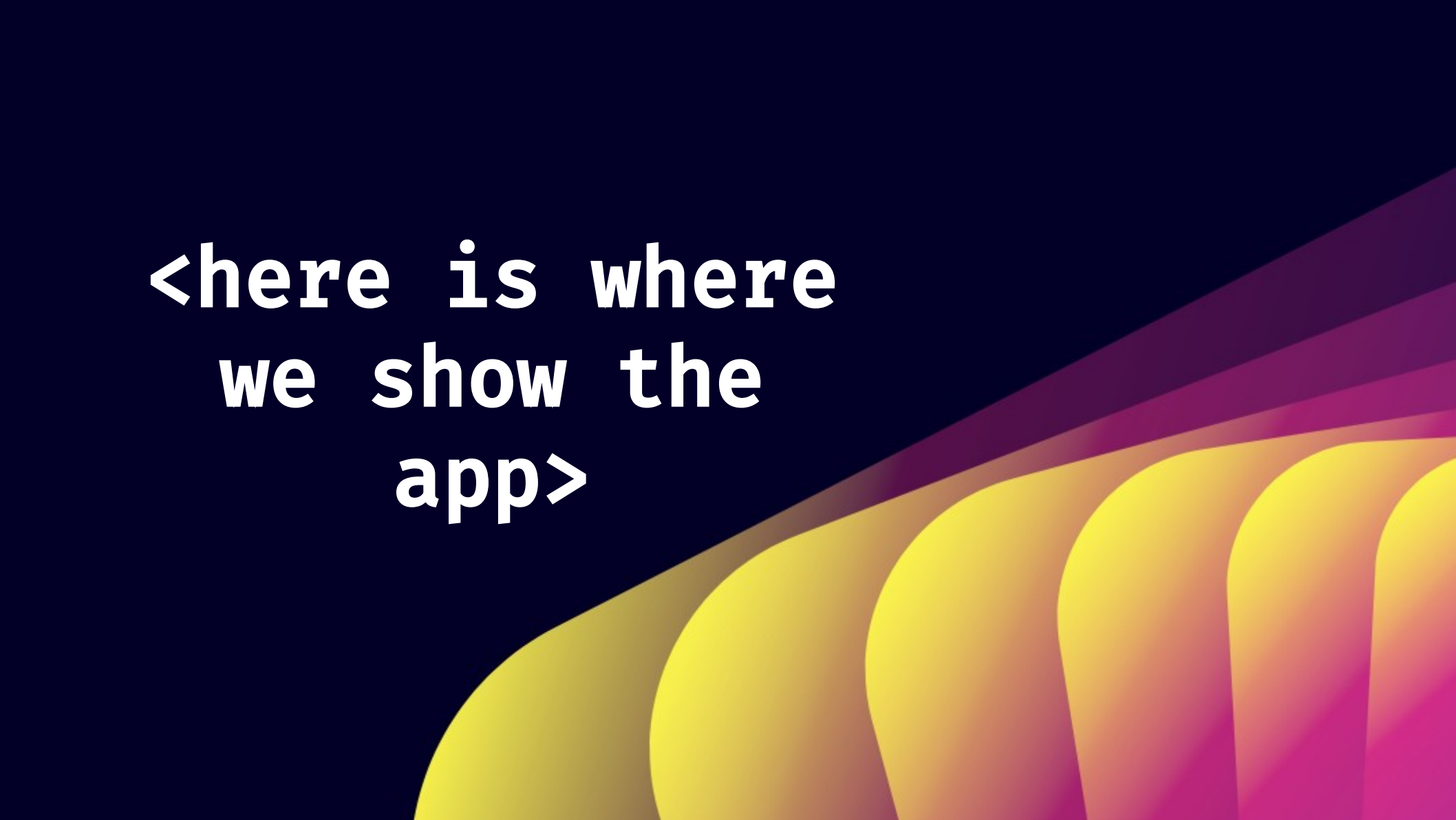
EVPanner.rs

*Dennis Orlando, Alessio Zeni,
Fabio Giovanazzi, Filippo Degrandi*

KONVERTO

passion for technology


**<here is where
we show the
app>**



Tech Stack



- Flutter (Android, Linux, Smart Fridge)
- OSRM, Nominatim
- Rust (actix-web & lots of crates)
- Docker Compose

Tech Stack

- Flutter (Android, Linux, Smart Fridge)
- OSRM, Nominatim
- Rust (actix-web & lots of crates)
...(blazingly fast )
- Docker Compose (docker compose up
and you're good to go)

Tech Stack

The screenshot shows the crates.io website interface. At the top left is the crates.io logo. A search bar contains the text "Type 'S' or '/' to search". To the right of the search bar are links for "Browse All Crates" and a user profile for "Degra". The main content area features the crate name "osrm_parser v0.1.0" and a "Follow" button. Below the name is the description "OSRM requests builder and response parser" and three tags: "#osrm", "#parser", and "#routing". A navigation bar includes "Readme", "1 Version", "Dependencies", "Dependents", and "Settings". The "Readme" section is active, displaying the title "OSRM Parser" and the word "Metadata".

crates.io Type 'S' or '/' to search  [Browse All Crates](#) |  **Degra** ▾

osrm_parser v0.1.0 [Follow](#)

OSRM requests builder and response parser

[#osrm](#) [#parser](#) [#routing](#)

[Readme](#) [1 Version](#) [Dependencies](#) [Dependents](#) [Settings](#)

OSRM Parser Metadata

Backend endpoints

```
#[post("/get_routes")]
pub async fn get_route(
    req: Form<PathRequest>,
    config: Data<AppConfig>,
    pool: Data<DbPool>,
) -> actix_web::Result<impl Responder> {
```

```
#[get("/get_all_stations")]
pub async fn get_all_stations(pool: Data<DbPool>) -> actix_web::Result<impl Responder> {
    let stations: Vec<StationInfo> = web::block(move || {
        let mut conn: PooledConnection<ConnectionManager<...>> = pool.get().unwrap();
        read_all_stations(&mut conn)
    })
    .await? Result<Vec<StationInfo>, ...>
    .unwrap();
    let stations: String = serde_json::to_string(&stations)?;
    Ok(stations)
}
```


**<here is where
we get back to
the app>**

The background features a dark blue gradient. In the lower right, there are several overlapping, rounded, wavy shapes in shades of yellow, orange, and purple, creating a sense of depth and movement.

Future Work

- Cost-optimizing algorithm
- Multiple appointments
- Minor bugs (i.e. politely arguing with OSM APIs)



Thank you! :)

*...and thanks to JetBrains for this
gorgeous Slide backgrounds <3*