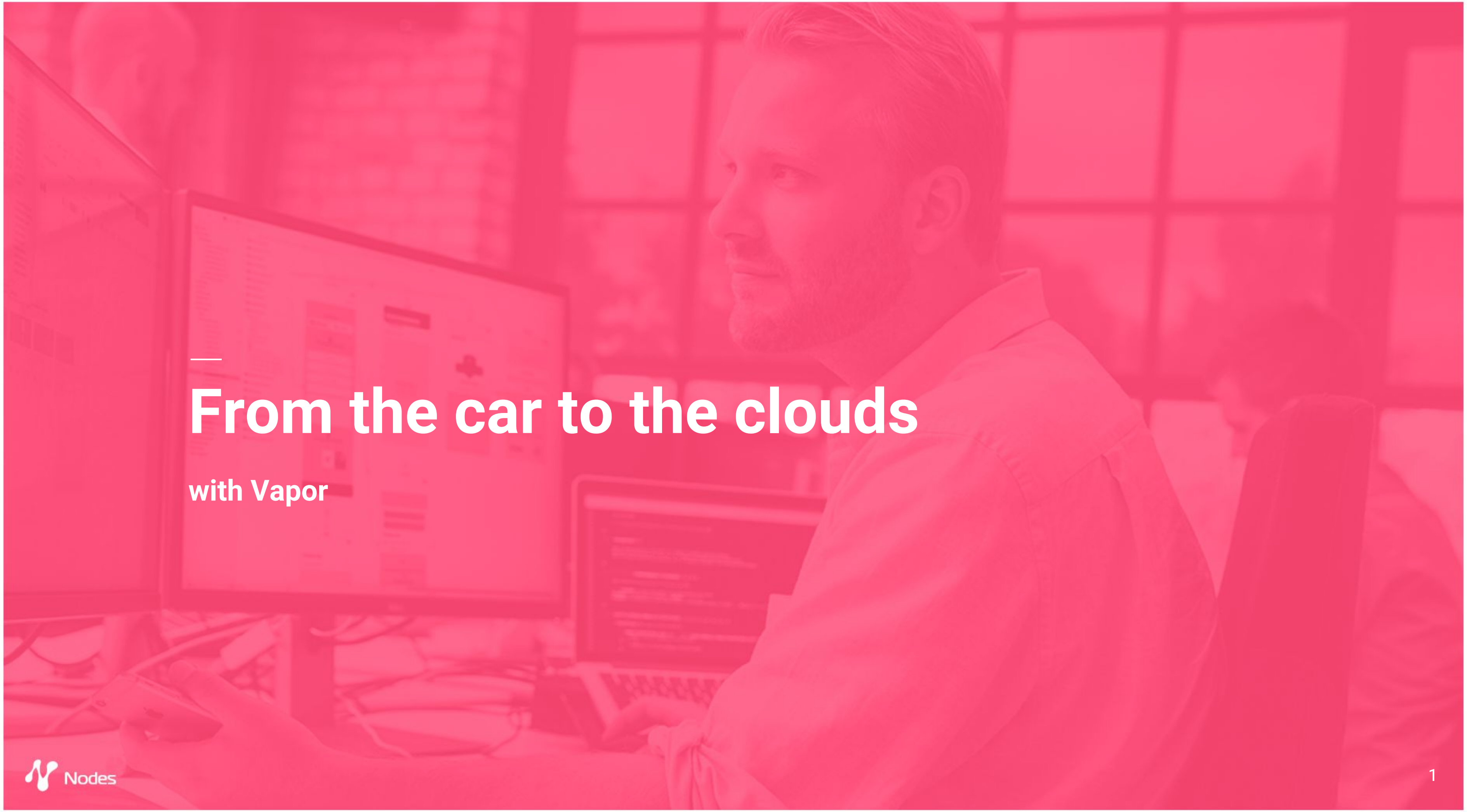


From the car to the clouds

with Vapor

Nodes

Pedro Coutinho

Senior Backend Developer - Nodes UK

php

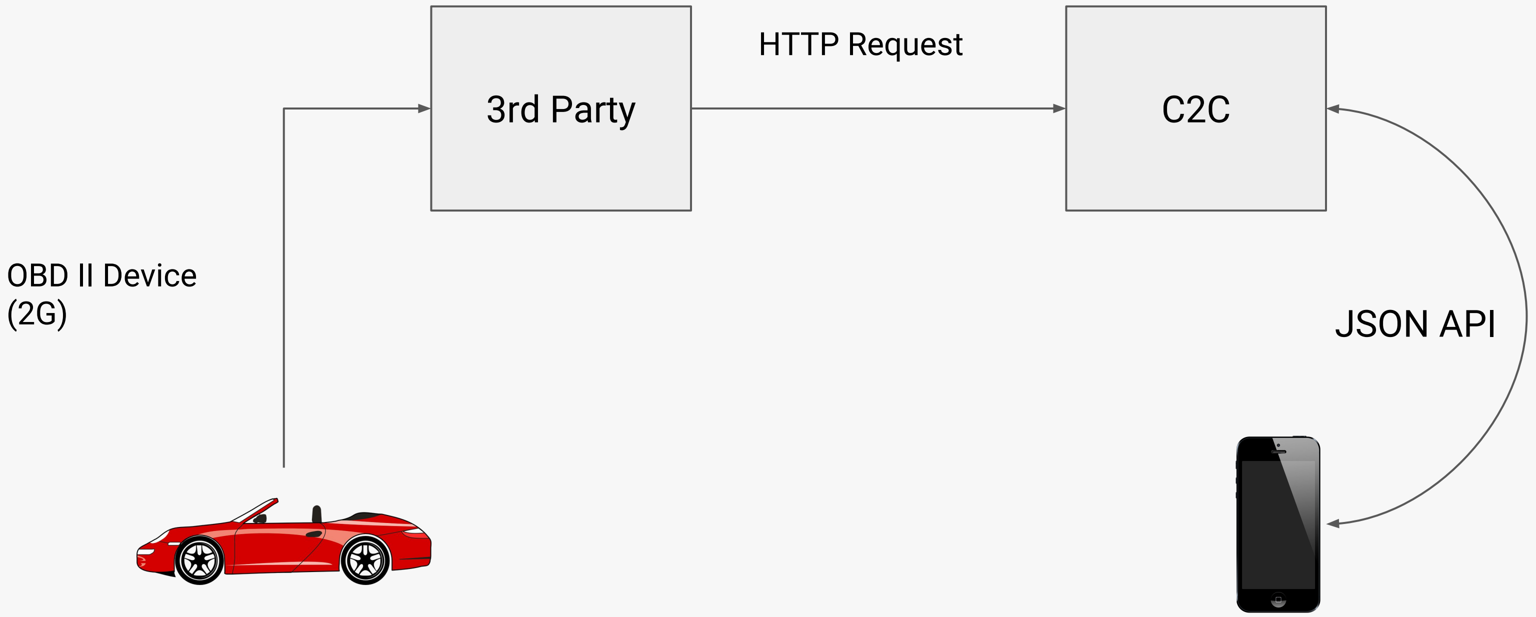
laravel

MySQL

C2C (Car to Cloud) - Initial Objectives

- Read driving data from 3rd party;
- Be able to group the events and define trips;
- Process the trip data to create scores;
- Send a push notification to the user when he finishes a trip.

Overview



Confidence level

- Piece of cake

- Clueless....

Where should I start?

Setting up

- Install Vapor and environment;
- Look for packages that may be useful;
- Checkout *The Swift Programming Language* book;
- Checkout some videos from Ray Wenderlich;
- Checkout Vapor slack channel;
- Read Vapor’s documentation.

Some of the tasks

- API User Management;
- Provide an API for the app;
- Integrate with 3rd party APIs.

API user management - JWT Keychain

I needed

- API User Management;
- Flexible out of the box solution.

Features

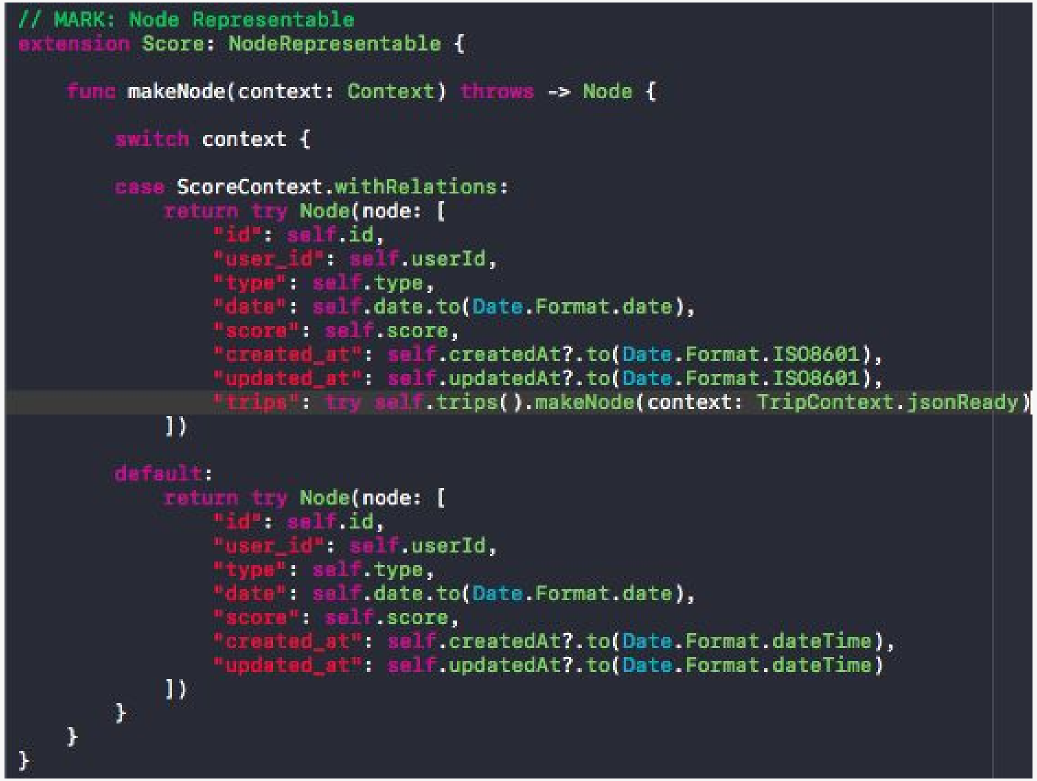
- User model ready to use (or make your own);
- User controller ready to use (or make your own);
- JWT token settings are configurable (expiry time, signer method, etc);
- Small set of routes that let you register, login, update info and retrieve your details;
- Reset password functionality (also configurable).

See more @ https://github.com/nodes-vapor/jwt-keychain

App API - Data transformation

- Usage of *.makeJSON()* was not enough, especially because of inner relations;
- Using an enumeration of Context on *.makeNode(context: Context)* allows the data to be converted in a flexible way (ex: model relations)

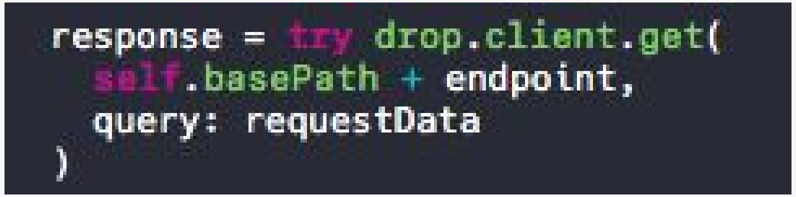
App API - Data transformation



```
// MARK: Node Representable
extension Score: NodeRepresentable {

    func makeNode(context: Context) throws -> Node {

        switch context {

        case ScoreContext.withRelations:
            return try Node(node: [
                "id": self.id,
                "user_id": self.userId,
                "type": self.type,
                "date": self.date.to(Date.Format.date),
                "score": self.score,
                "created_at": self.createdAt?.to(Date.Format.ISO8601),
                "updated_at": self.updatedAt?.to(Date.Format.ISO8601),
                "trips": try self.trips().makeNode(context: TripContext.jsonReady)
            ])

        default:
            return try Node(node: [
                "id": self.id,
                "user_id": self.userId,
                "type": self.type,
                "date": self.date.to(Date.Format.date),
                "score": self.score,
                "created_at": self.createdAt?.to(Date.Format.dateTime),
                "updated_at": self.updatedAt?.to(Date.Format.dateTime)
            ])
        }
    }
}
```

3rd party APIs - Vapor’s HTTP Client

- Native Vapor Client;
- Made the integration with Google Maps services a breeze;

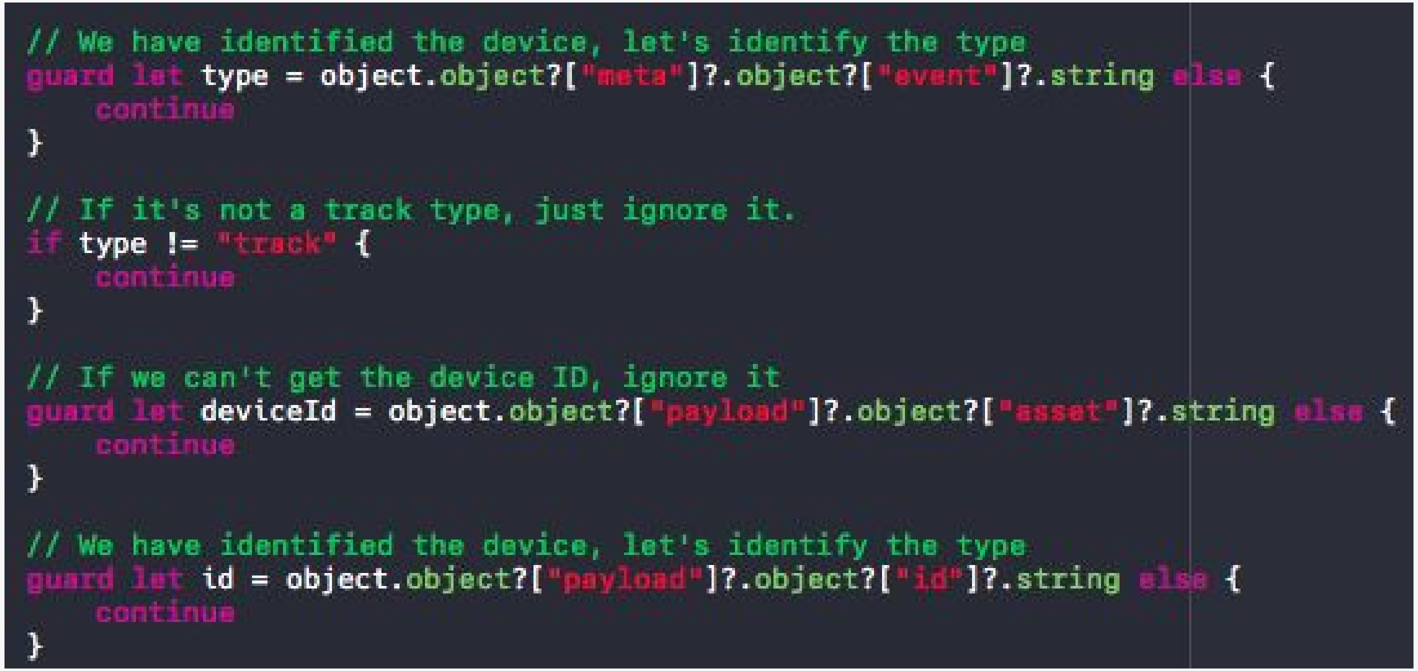


```
response = try drop.client.get(
  self.basePath + endpoint,
  query: requestData
)
```

3rd party APIs - The Optionals Pain

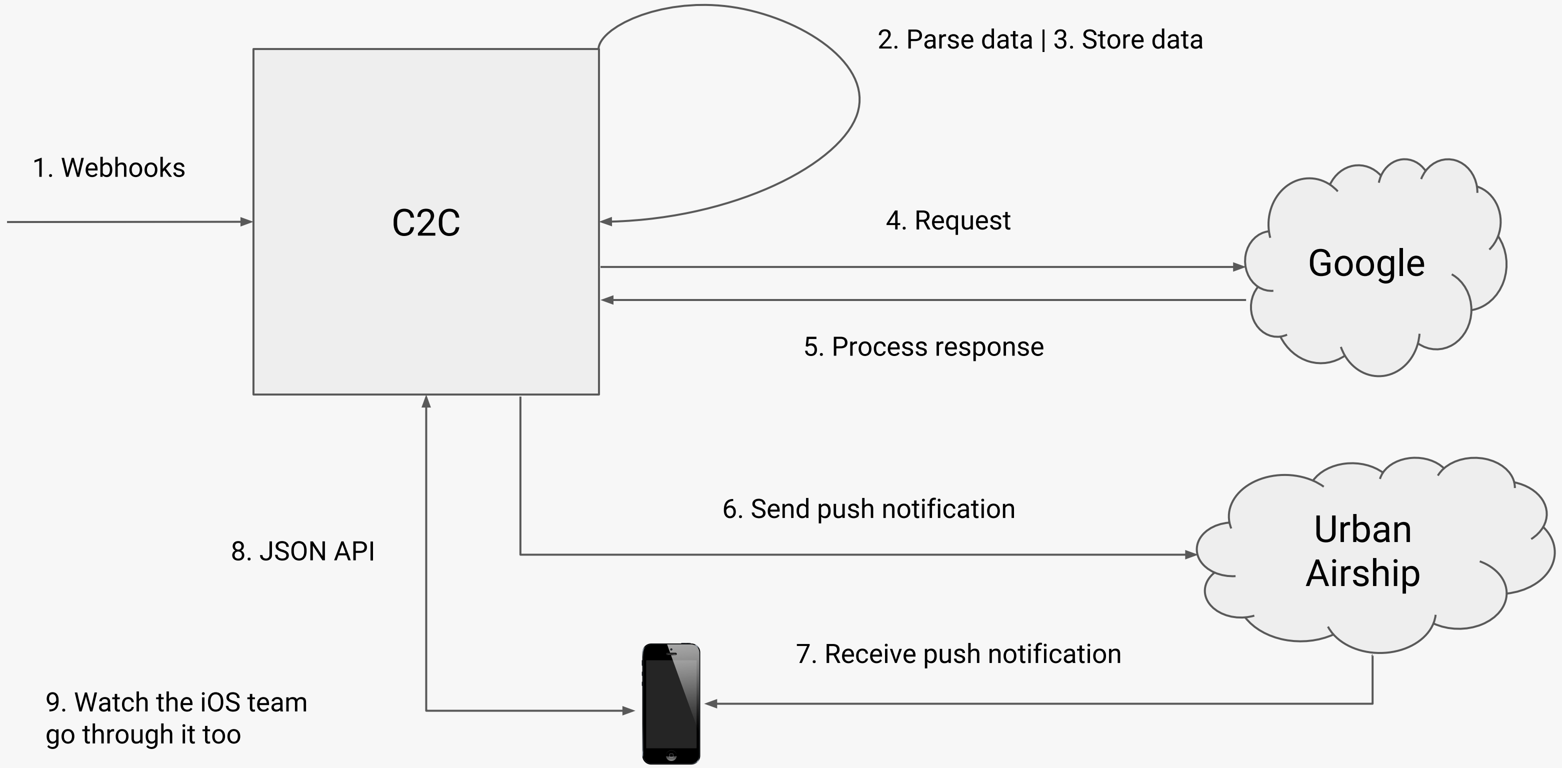
- Very painful to deal with coming from the PHP world;
- Parsing the JSON webhook from the 3rd party was a big lesson on how important and good they are;

3rd party APIs - The Optionals Pain



```
// We have identified the device, let's identify the type
guard let type = object.object?["meta"]?.object?["event"]?.string else {
    continue
}

// If it's not a track type, just ignore it.
if type != "track" {
    continue
}

// If we can't get the device ID, ignore it
guard let deviceId = object.object?["payload"]?.object?["asset"]?.string else {
    continue
}

// We have identified the device, let's identify the type
guard let id = object.object?["payload"]?.object?["id"]?.string else {
    continue
}
```

The optional pain scheme


2. Parse data | 3. Store data
1. Webhooks
C2C
4. Request
Google
5. Process response
6. Send push notification
Urban Airship
8. JSON API
7. Receive push notification
9. Watch the iOS team go through it too

With Laravel it would be a piece of cake, but...

It turned out that Vapor was a piece of cake too!

- 2 weeks to a fully working prototype;
- Easy to understand coming from Laravel;
- An awesome community behind Vapor.

Thank you