Exploring React Native & Expo: Architecture and Implementation Feasibility

Internship Presentation
Mina Khosh Nazar
October 2024

Agenda

Introduction to Ecosteer

Manual Preparation

Research on Necessary Packages

Package Testing

Development of Mini Technical Document

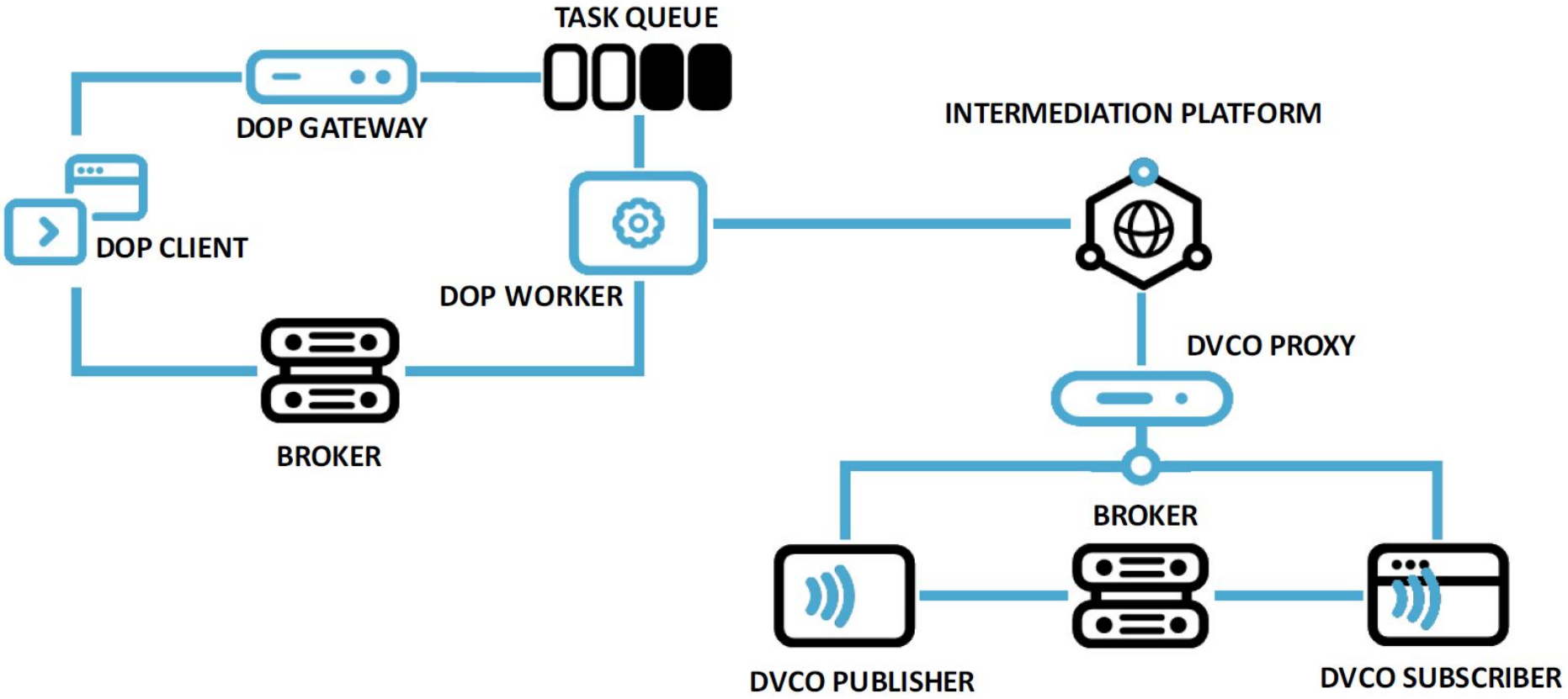
Ecosteer Architecture Overview

DOP (Data Ownership Platform)

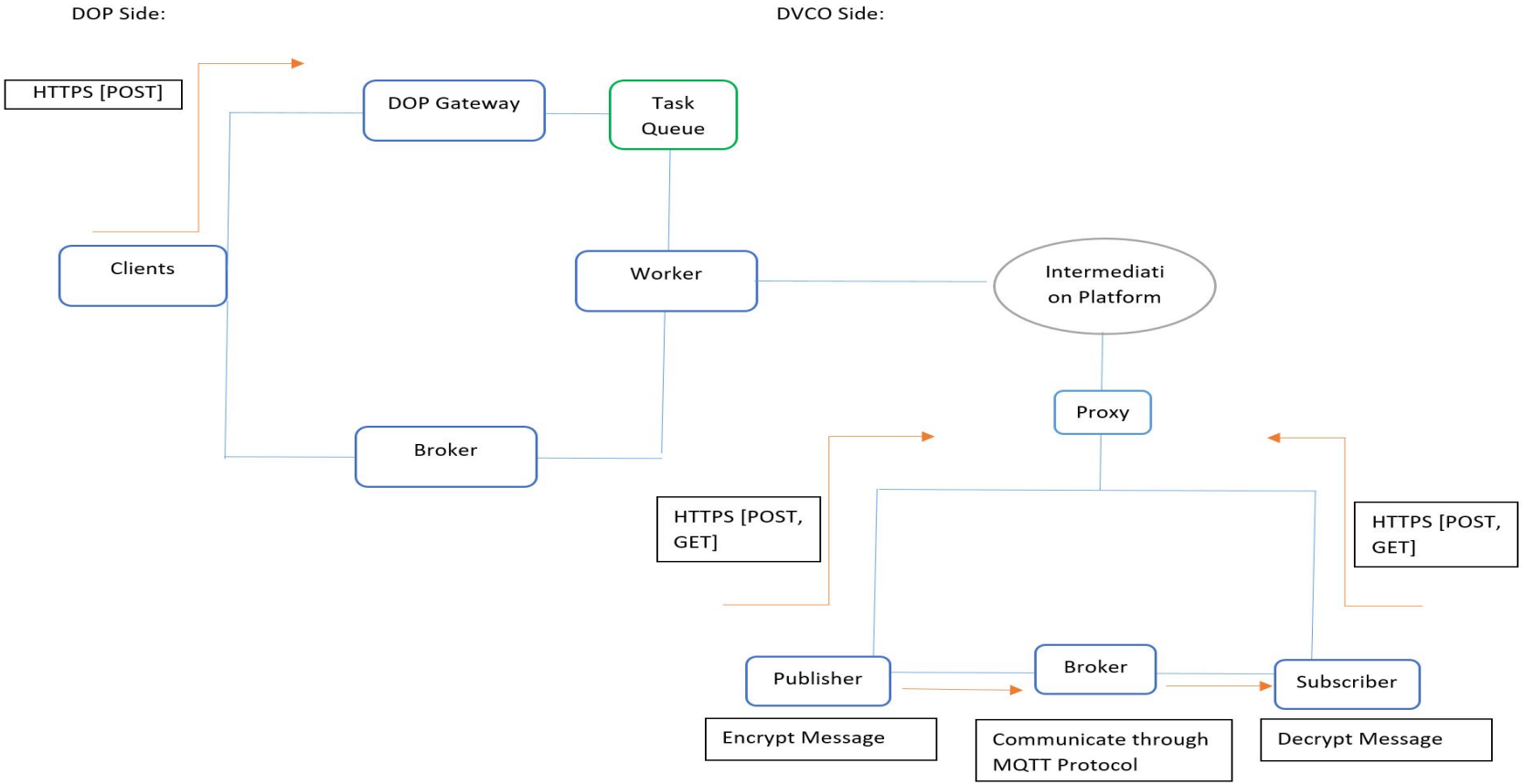
- Overview of core functionalities and architecture
- Sending HTTPS Requests to the Gateway
- Mechanism for secure data transfer to the DOP Gateway
- Implementation details for HTTPs communication in DVCO

DVCO (Data Visibility Control Overlay)

- Focus on **Pub/Sub and Broker Setup**
  - Configuring the publish/subscribe model within DVCO
  - Understanding broker roles and operations


TASK QUEUE
DOP GATEWAY
INTERMEDIATION PLATFORM
DOP CLIENT
DOP WORKER
DVCO PROXY
BROKER
BROKER
DVCO PUBLISHER
DVCO SUBSCRIBER

Road Map to React Native Mobile App



User Manual for React Native & Expo

The manual includes of seven main sections, outlined as below:

- Installing the Development Environment
- Creating First React Native Project
- Understanding the Project Structure
- Modifying the App
- Network Configuration

User Manual for React Native & Expo

Installing the Development Environment

- Step-by-step guide for setting up Node.js, Expo CLI, and other essentials.

Creating First React Native Project

- Instructions to create and launch a new project with Expo.

Understanding the Project Structure

- Overview of key files and folders in a React Native project, explaining their purposes.

Modifying the App

- Basic customization steps to edit components and see live updates.

Network Configuration

- Setting up network access, including configuring LAN settings and dealing with IPs for mobile device access.

Creating an Expo App: Recommended Approach

Use `npx create-expo-app projectName`

- This command is recommended as it provides a well-organized structure, supports TypeScript by default, and aligns with current best practices for Expo development.

Avoid `expo init projectName`

- The traditional Expo CLI is deprecated, making this command less suitable for new projects.

Necessary packages

Axios

- Used for sending HTTPs requests (GET and POST) to interact with APIs and servers.

mqtt.js

- Manages the Mosquitto broker for efficient publish/subscribe messaging.

Websockets

- Enables secure, real-time communication between the app and backend services.

crypto-js

- Provides encryption and decryption functionalities to secure data transmission.

Polyfills: Extending Compatibility

- Why Polyfills?
  - In this project, polyfills are essential to bridge compatibility gaps between React Native and certain JavaScript libraries or modules.
- We specifically need polyfills in this project to:

- Ensure compatibility with Node.js-based packages, such as **mqtt.js**, which require certain environment features not natively supported in React Native.

- Support cryptographic operations, ensuring that encryption/decryption processes can be seamlessly executed on mobile devices.

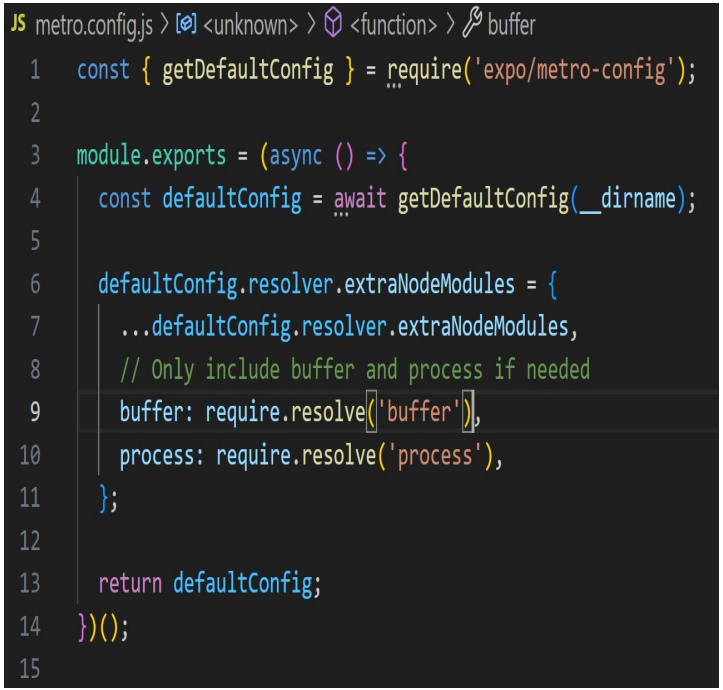
Polyfills: Extending Compatibility

Implementing Polyfills

1. Using **`metro.config.js`** File:
   - Create a “metro.config.js” file with the necessary configuration.
   - Import polyfills in your target component for compatibility.

e.g:

import 'react-native-url-polyfill/auto';



Polyfills: Extending Compatibility

Implementing Polyfills

- Simply import polyfills directly into the component where they are needed, without configuring “metro.config.js”.

e.g:

```
import 'react-native-get-random-values';
```

This method makes the polyfill available in the component where it's used.

Difference between Polyfills Methods

1. “metro.config.js”:

   - Polyfills are applied globally.

   - Suitable for app-wide usage.

2. Direct Import:

   - Polyfills are imported only in specific components.

   - No “metro.config.js” changes needed.

   - Best for localized use.

In summary, use “metro.config.js” for global polyfills, and direct imports for component-specific needs.

Polyfills: Extending Compatibility

- What Polyfills are used in this project?

- URLs, Buffer and Process (mqtt.js library)

- Using random value for pseudorandom generator key (crypto-js library)

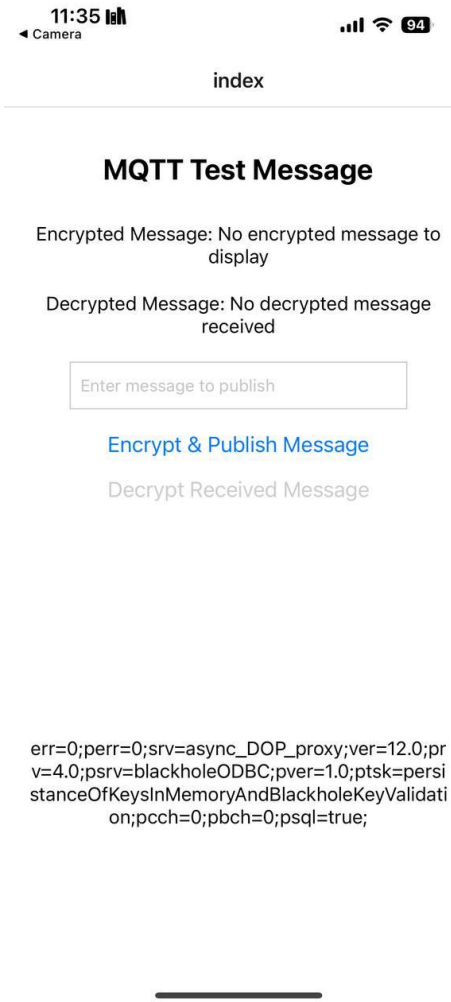
Tests

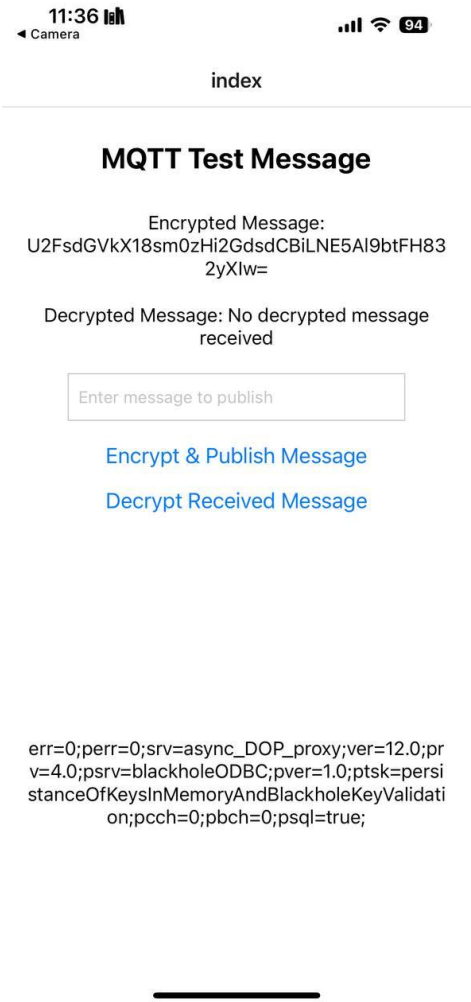
Project Setup:

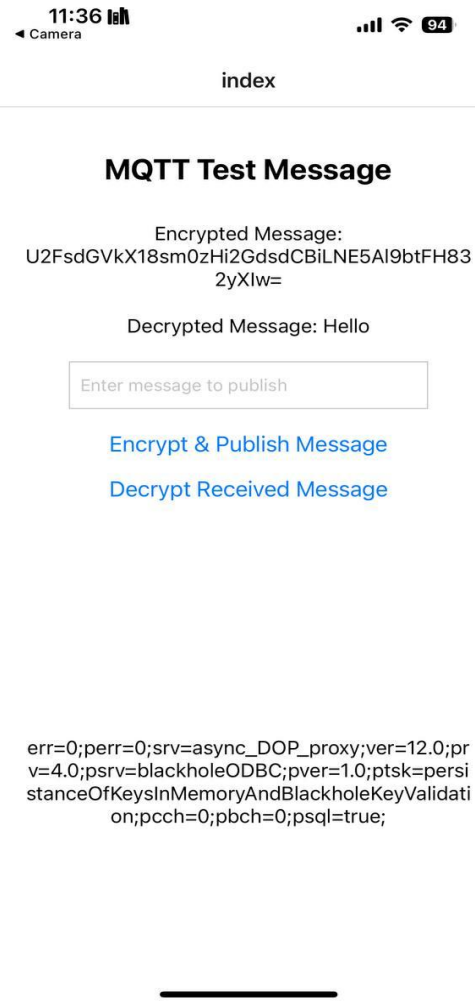
- A simple project was created using `npx create-expo-app final` to test package integration.

Result:

- App connects to the proxy
- Displays **JSON response** at the bottom of the page
- Connects to **Mosquitto broker**
- Publishes encrypted messages
- Subscriber **decrypts** and displays the original message in the app


11:35
Camera
94
index
MQTT Test Message
Encrypted Message: No encrypted message to display
Decrypted Message: No decrypted message received
Enter message to publish
Encrypt & Publish Message
Decrypt Received Message
err=0;perr=0;srv=async_DOP_proxy;ver=12.0;prv=4.0;psrv=blackholeODBC;pver=1.0;ptsk=persistanceOfKeysInMemoryAndBlackholeKeyValidation;pcch=0;pbch=0;psql=true;


11:36
Camera
94
index
MQTT Test Message
Encrypted Message: U2FsdGVkX18sm0zHi2GdsdCBiLNE5Al9btFH832yXIw=
Decrypted Message: No decrypted message received
Enter message to publish
Encrypt & Publish Message
Decrypt Received Message
err=0;perr=0;srv=async_DOP_proxy;ver=12.0;prv=4.0;psrv=blackholeODBC;pver=1.0;ptsk=persistanceOfKeysInMemoryAndBlackholeKeyValidation;pcch=0;pbch=0;psql=true;


11:36
Camera
94
index
MQTT Test Message
Encrypted Message: U2FsdGVkX18sm0zHi2GdsdCBiLNE5Al9btFH832yXIw=
Decrypted Message: Hello
Enter message to publish
Encrypt & Publish Message
Decrypt Received Message
err=0;perr=0;srv=async_DOP_proxy;ver=12.0;prv=4.0;psrv=blackholeODBC;pver=1.0;ptsk=persistanceOfKeysInMemoryAndBlackholeKeyValidation;pcch=0;pbch=0;psql=true;

```
Logs for your project will appear below. Press Ctrl+C to exit.
(node:14240) [DEP0044] DeprecationWarning: The `util.isArray` API is deprecated. Please use `Array.isArray()` instead.
iOS Bundled 113509ms D:\Unibz\test\final\node_modules\expo-router\entry.js (1242 modules)
 LOG  Connected to MQTT broker
 LOG  Subscribed to topic
 LOG  Encrypted message: U2FsdGVkX18sm0zHi2GdsdCBiLNE5Al9btFH832yXIw=
 LOG  Encrypted message published successfully: U2FsdGVkX18sm0zHi2GdsdCBiLNE5Al9btFH832yXIw=
 LOG  Encrypted message received on topic: U2FsdGVkX18sm0zHi2GdsdCBiLNE5Al9btFH832yXIw=
 LOG  Decrypted message: Hello
```

Mini Technical Document

In-depth Guide to Programming with React Native:

- Explanation of polyfills and their role in enhancing compatibility within React Native.
- Overview of React Hooks and how they streamline state management and lifecycle methods.
- Best practices for organizing imports to avoid redundancy and improve code readability.
- Tips on setting up a clear project structure and leveraging StyleSheet for organized styling.

Challenges

Understanding the Ecosteer Architecture

Connecting Phone to The Project Through a Public Network

Implementing Encryption with Random Number Generation (seed)

Next Project: Mobile App Development with DOP API

Using this knowledge to create a mobile application which works with DOP API.

- **Data Owner's App Development**:
  - Built using **React Native**
  - Key features:
    - View published assets
    - Manage subscriptions
    - Control visibility of data streams (grant/revoke access)
- **DOP API Integration**:
  - Seamless integration with DOP APIs
  - Focus on handling backend **event notifications**

Thank You!