ARCHITECTURAL DESIGNS

WILDLIFE DRONES AND FLIGHT PLANS

Stakeholders

Matthew Evans
Reinhardt Eiselen
Brayan Jansen van Vuuren
Andreas Louw
Deane Roos

Team DR BAM
drbam301@gmail.com

Contents

System architectural design

The overall system follows a client-server architecture. The system requires a high level of security and secrecy due to the nature of poaching, and, as such, all processing will be done server side (otherwise the client could be reverse engineered). Anti-poaching data is considered very sensitive, and this method allows all data to be stored server side, and only results of computation to be provided client side.

Most communication between the client and server uses the HTTPS protocol. This provides a simple and secure protocol catered to requests and responses – perfect for the client to call the API. Sockets will be used to communicate live data between the client and server – such as current device geolocation.

The client-server architecture also provides a relationship where the clients need not know each other – this enables the client to be written in a multitude of languages/frameworks for different operating systems, and they will all work so long as they adhere to the interface of the server. This will make cross-platform development easier.

Another advantage to client-server architecture is processing can be offloaded from the client devices to the server, which will typically be mobile devices where battery life is critical to the user. This will result in a better user experience. A drawback to this approach is that the server and client need to have constant communication in order to work – as such, a caching layer is implemented to provide limited offline functionality to the client in the case of network or server error (discussed later).

On the server side, an N-tier architecture is used with Model View Controller. The layers are implemented as a connection pool, middleware, routes, and a persistence layer. This provides a modular architecture where parts of the system are interchangeable and easily modified thanks to low coupling between the layers. The connection pool is handled by Node.js and Express.js, providing an efficient server that can handle multiple concurrent clients. The middleware is handled by Nest.js, and provides an abstraction over the API endpoint routes to perform reusable actions such as authentication and error handling.

The server follows MVC, because the model is realised as a persistence layer, the controller as routes and services, and the view as the API's JSON REST interface. Services are associated with controllers via dependency injection, making them reusable and easily replaceable (low coupling). Services can therefore be designed with high cohesion in mind, adhering to the single responsibility principle.

Client-side, Model View Controller is also used in an N-tier architecture. The view is an HTML and CSS representation rendered using Cordova or the web browser. The controller is realised using

controllers and services. As with the server side, dependency injection is used and provides the same benefits as for the server. The model is implemented as a cache between server calls and a persistence layer.

Subsystems

Map System

The map subsystem will be an interactive subsystem where the user will have to login to the system, this will allow the user access to the map. The n-tier architecture style will be used.

Hotspot identification subsystem

The hotspot identification subsystem will be a heuristic problem-solving system where the hotspots that will be identified will use various data from the system these include but not limited to: past poaching incidents, probability of animals in area, probability of poaching, interest points allocated by users. This data will be used to predict possible poaching hotspots that may occur in future. The blackboard architectural style will be used.

Animal prediction subsystem

The animal prediction system will be a heuristic problem-solving system where the location of an animal will be predicted for an x amount of time in the future. The future location of the animal will be predicted using varies data that include but is not limited to: past location, habitat, temperature, weather, distance to water. The blackboard architectural style will be used.

Route generation System

Route generation subsystem will be an interactive subsystem where the user will initiate a new route to be generated, the system will respond with the new route shown on the map. The N-tier architectural style will be used.

Data modification subsystem

The data modification subsystem will be an interactive and an object-persistence system; it is an interactive system by allowing the user(administrator) or other subsystems to add, remove and modifying data that is stored in the database this will have to be initialised by the user to start the process.. The system is also an object-persistence system by allowing the for storing and retrieving objects from the database and file systems. The system will mainly use n tier architecture with a persistence framework architecture on a lower tier of the n tier architecture.

Geolocation subsystem

The geolocation subsystem will be an event-driven system as a device (smartphone or drone) will ping the system and send the current location to the system where the system will then relay the data to the server. An event-driven system architecture will be used.

Notification subsystem

The notification subsystem will be an event-driven system that will be triggered by other systems such as the data modification, route generation and other systems by sending the notification system data and the notification system will then process the data accordingly. An event-driven system architecture will be used.

Authentication subsystem

The authentication system will be an interactive system as the user has to interact with the system to log in the data will then be processed by the authentication system to either allow access or deny access. A n-tier architecture will be used.

Reporting subsystem

The reporting system will be an event-driven system as other system will send data to the reporting system, the system will then process the data and send the data to the data modification system. An event-driven system architecture will be used.

Deployment Model

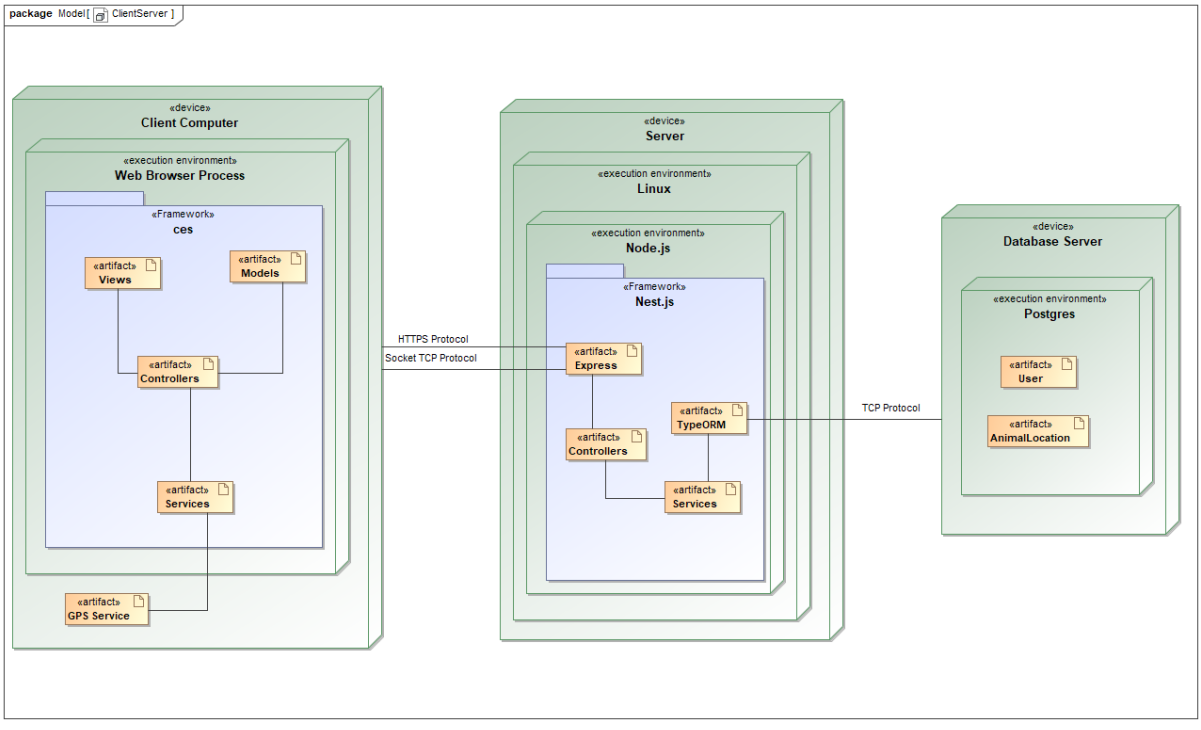

Figure 1 Deployment model