Final Report

Project Nest

Massey University Capstone 2016 - Group Two

Team Members:

Zoe Purdon-Udy (13047928)

Francis Greatorex (14136398)

Sam Hunt (14216618)

MJ Lee (13214654)

Introduction

Project Nest is the name coined for group two's efforts in producing a mobile application in fulfillment of the requirements of Massey University's software engineering capstone project. Project Nest also encompasses the web application deployed to the world wide web, and the backend developed to support these two consumer products.

The following report documents the project as it exists at the final submission date of the 31st of October, 2016.

Contents

Project Requirements

The requirements, as defined in the project description set by 159.356 paper coordinator

Essential Requirements

All of the following requirements have been achieved/implemented

- ➢ supports iOS and Android
- ➢ should support operating systems from 2013 onwards
- ➢ mobile app. can be used to locate traps using acoustic and visual signals
- ➢ master data for a particular trapline can be configured
- ➢ an instance of the app can be protected by a password
- ➢ mobile app. can be used to capture data
- ➢ data can be synchronised with a central data store
- ➢ captured data can be exported

Optional Requirements

The following requirements were optional requirements that have also been implemented

- ➢ web application facilitates admin actions
- ➢ admin access to web is secured using user accounts with password protection

Architecture and Design

Architecture and design diagrams and explanations that are consistent with the final product

High Level Design

The main components of the software are isolated from one another through the use of the JSON data format and the HTTP 1.1 transmission protocol. HTTP and JSON are text-formatted and independent of single specific technology, therefore many languages have the tools to interpret them. The use of HTTP and JSON allowed the choices for web technologies and the mobile development language/environment to be chosen independently of each other.

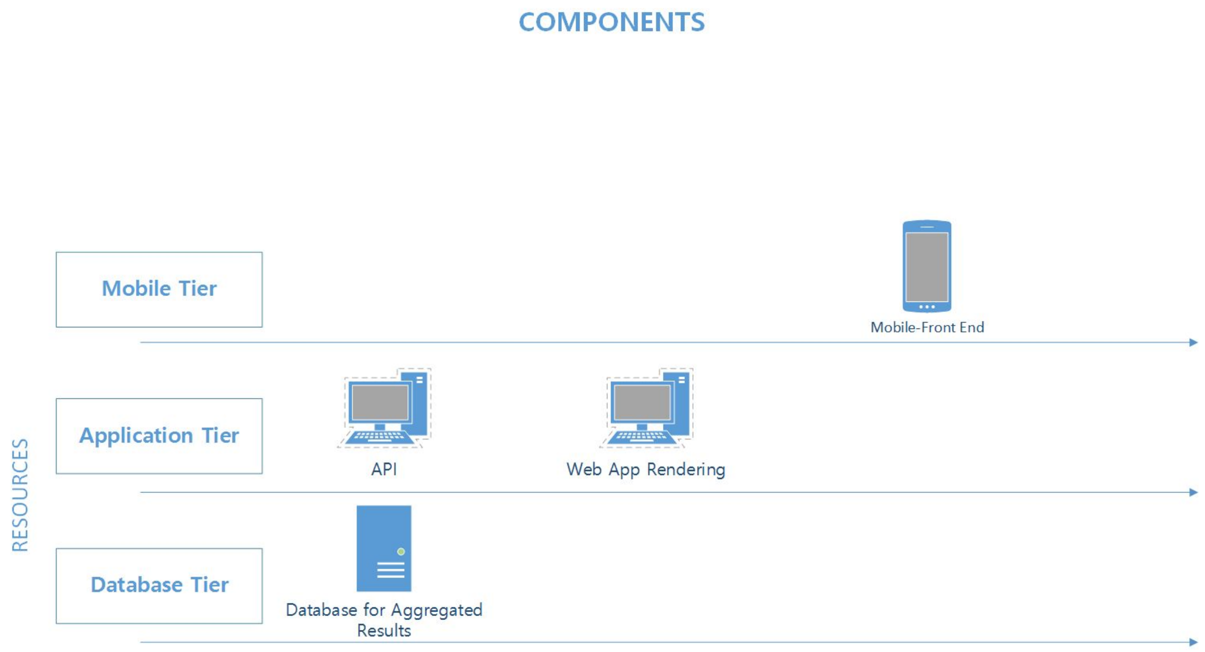
Separating the API between EC2 and RDS (both could have been hosted on the EC2) demonstrates that it is fine if the database and request handler are on separate machines.

RESTful API Specification

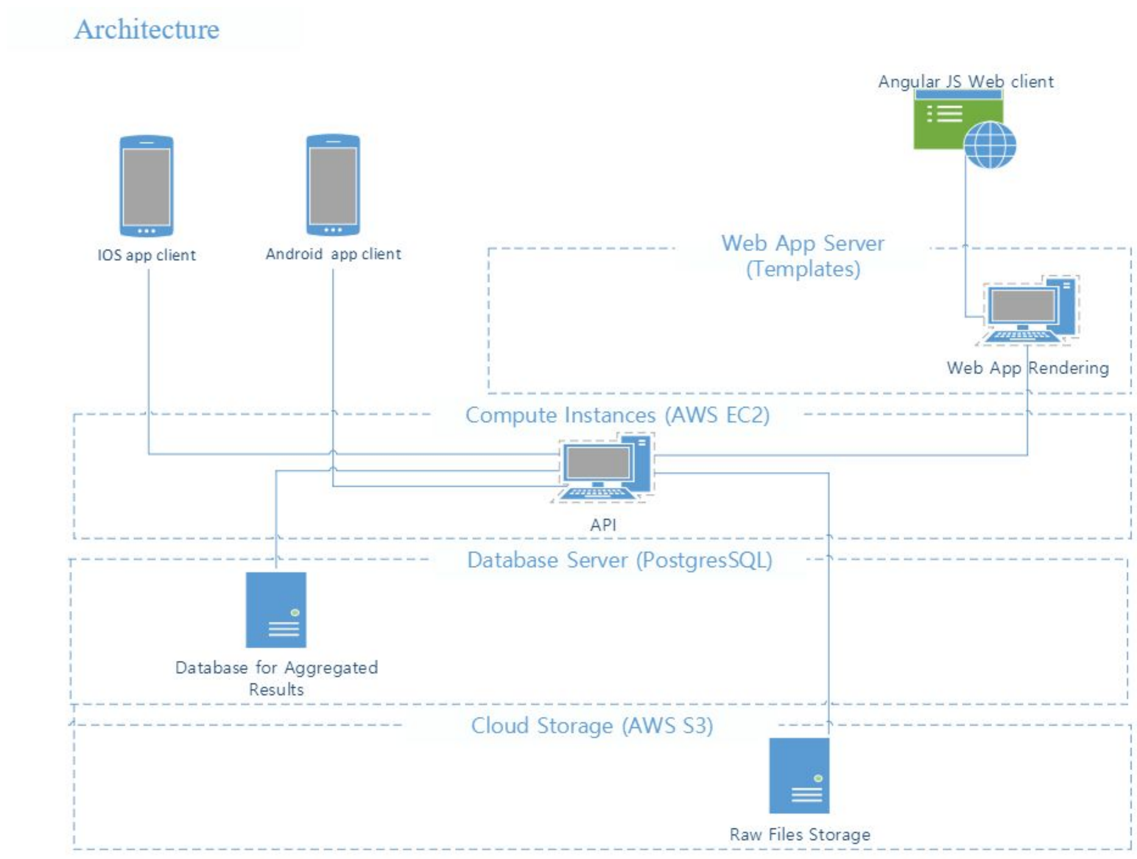
The complete REST spec document is attached as an appendix to this report

The REST specification contains details about all communication protocols used between components.

Component Design

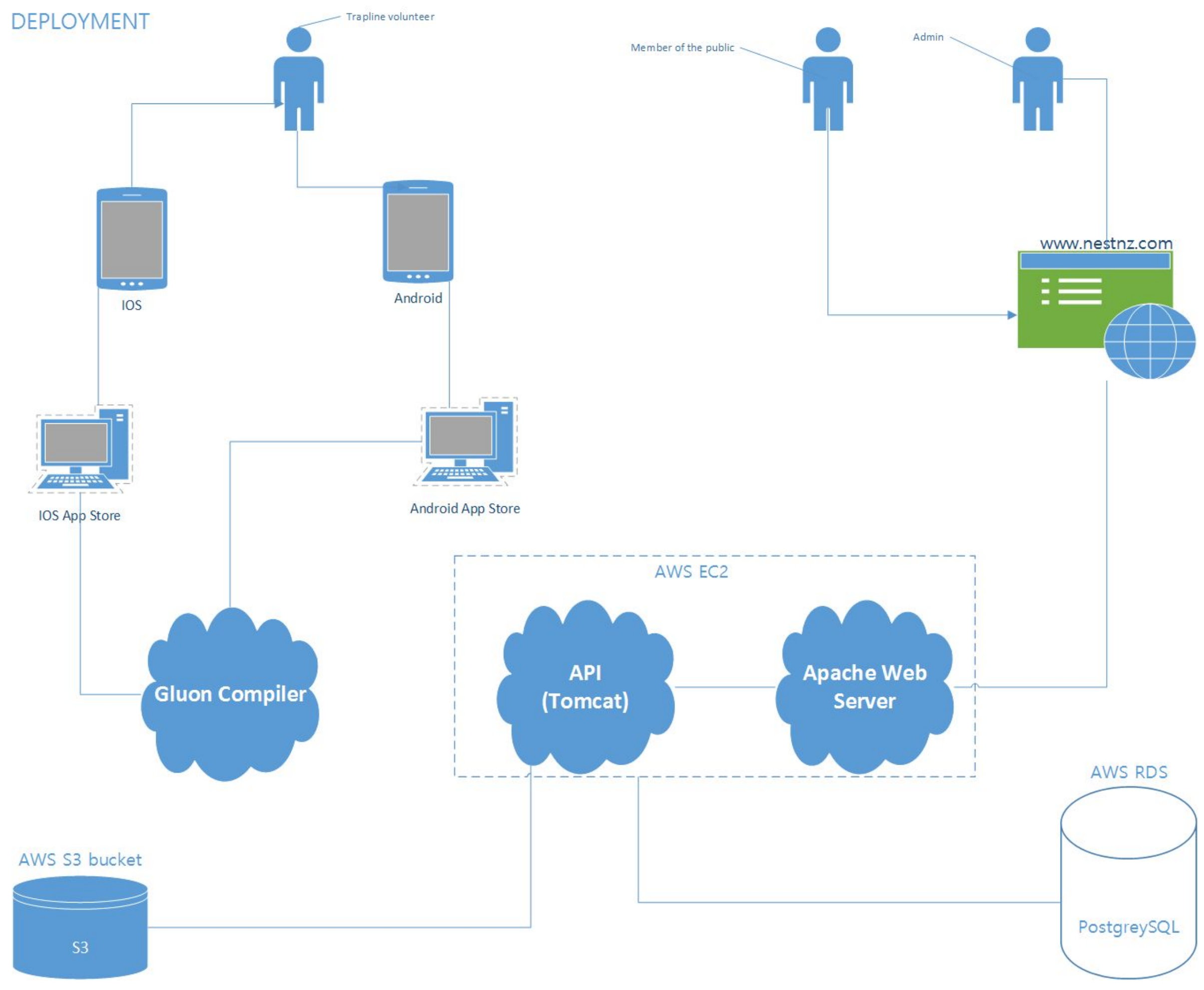


Architecture Design



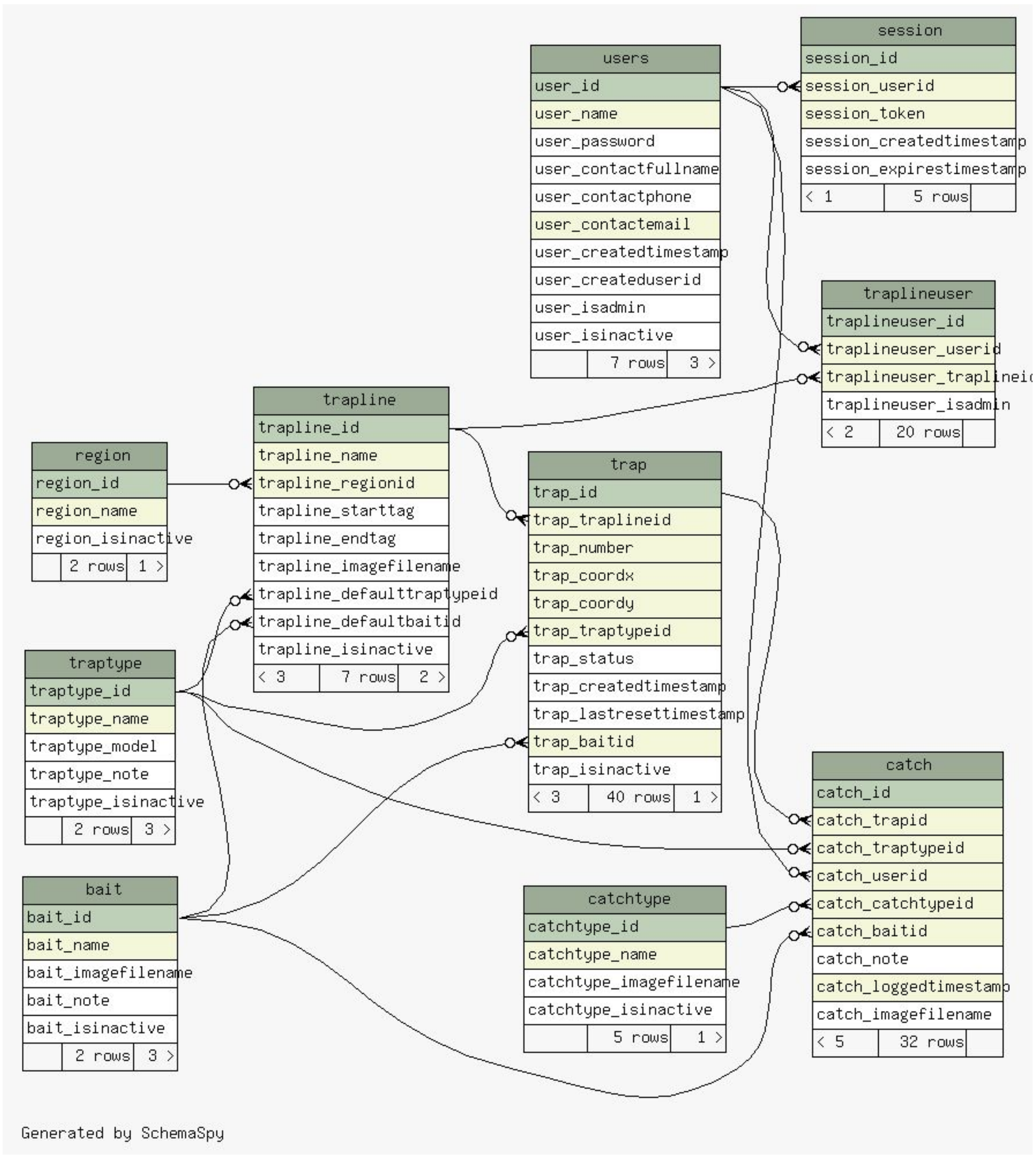
The above architecture design shows the interactions between the components. API is located in the middle of the components to provide interaction services to other components.

Deployment Diagram



Database Design

Generated using SchemaSpy from our production database



Key:

Darkest green (top row) - table name
Mid green (second top row) - table primary key column (clustered index)
Yellow - indexed columns (non-clustered indexes). Includes:

- Foreign keys (indexed for performance),

- Constrained columns (Postgres implementation for uniqueness and conditional check constraints etc.)

White - non-indexed columns.

Final row (from left to right) -

a). number of parent table relationships or number of entities the table depends on
b). current number of rows in the table (snapshot from time of diagram generation)
c). number of child table relationships or number of entities which depend on it

Note:

- the ER diagram is arranged topographically from left to right, with tables higher in the hierarchy on the left
- crow's foot notation on arrows indicates a one to many relationship between the various foreign key columns
- relationship arrows are connected to the appropriate foreign key columns on the tables which contain them

Quality Assurance

Test coverage metrics and other QA reports

Mobile Application

PMD

As shown in the PMD report (Appendix A), no issues were found in the code analysed. However, several classes were unable to be analysed due to an issue (possibly due to not supporting Java 8, like JDepend)

JDepend

The JDepend report (Appendix B) only takes account of classes which don't use lambda expressions, as JDepend has not been updated to support the "invokedynamic" opcode used in Java 8 to create lambda methods. Running JDepend as part of "gradle check" causes a series of these messages to be logged in the console:

```
[ant:jdependreport] Unknown constant: 18
```

Because of this, the metrics from the report are not reliable.

JaCoCo

The Java Code Coverage JaCoCo) report (Appendix C) shows a test coverage rate of about 30%. Neither the UI elements nor network request elements were tested via unit testing (which the attached report covers). Since these elements make up a significant portion of the code base, the overall test coverage is fairly low, though the areas which are tested are well covered.

The UI elements are affected by factors outside the control of the app's code (i.e. device size, platform, internal code for the JavaFX/Gluon libraries), so it doesn't make sense for them to be unit tested. Instead, we have performed usability testing on several different types of devices to make sure all elements behave as expected.

Similarly, network elements require interaction with a server, which puts them outside the scope of unit tests. Instead, system tests have been performed where the whole app is used where functionality (including network elements) has been tested. Like the usability tests mentioned above, these will not show up in code coverage metrics.

Web Application

Selenium IDE is a Firefox plugin which is used to record user actions, and to export them as a reusable test script which imitates the user behaviour. The tests in the IDE are limited to Firefox, however, this limitation of Selenium IDE can be overcome by using WebDriver in conjunction with the exported JUnit tests. WebDriver automates browser compatibility testing by launching the various web browsers from JUnit. This forms the basis of regression testing for the web component.

API

The API request handler fully implements activity logging, and currently has 11 test classes, implementing 80 automated unit test methods using JUnit 4.12

Logging

As the API request handler runs in the tomcat container, all requests and HTTP activity are logged in the tomcat log. Tomcat uses an extended implementation of java.util.logging called 'JULI' which has better concurrency support, etc:
https://tomcat.apache.org/tomcat-7.0-doc/logging.html

Having access to the full-text of the HTTP request, as well as the full-text of the SQL query generated by the handler was invaluable for debugging integration issues between the web application and mobile application and the API.

Testing

A test-server and test-database were set up on the API development machine, so as not to break the production system (for integration testing) while development was ongoing on the API. Configuration was abstracted so that code pushed from the test system to production, would instead use the production configuration (login params etc) and database.

Manual HTTP testing was done using the Postman HTTP client (cURL + nice GUI):
https://www.getpostman.com/

Database testing and management was done using pgadmin3:
https://www.pgadmin.org/

White box unit tests check that:

- Internal functions succeed/throw exceptions as expected
- All API config files exist and contain the parameters required
- All mapped URLs have syntactically valid and parsable datasets
- Database config allows successful connections
- The database interface layer correctly casts types into and from the db

The API was faced with a large obstacle to testing, in that a large volume of the logical code is actually located outside of the core Java request handler.

This is due to the 'dataset' model used by the API. The Java core mainly acts only as an interface layer between the clients and the database. Functionally, it translates HTTP requests into SQL using templates and sends it to the database, while having no actual knowledge of the database structure itself, other than how to connect to it.

For this reason, that most of the logic for accessing the database was stored in the SQL templates, (in the ProjectNest-API/WEB-INF/datasets folder), black-box style test cases were required. Assuming that:

The test API server is running on localhost:8084 and
The test database is running on localhost:8432,

test cases made HTTP requests to the API, and the responses were check for expected states. Any records created during testing were tracked and removed transparently so that the tests could run on a live system without interrupting it.

Project Statistics

Statistics from the project's code repository, the Github Issue Tracker, and the team's communication channels

Github

Issues classified by which component of the project they affect

- App
- Web
- API

Issues tagged with their category

- Bug
- Enhancement
- Question (meaning point for discussion)
- Core
- Future Development
- Usability
- Report/Presentation

Patterns and trends in the issue tracker

Total commits: 723
Branches: 5

Team member	Issues opened	Assigned issues (open)	Assigned issues (closed)
Francis	77	7	65
MJ	35	3	37
Sam	66	4	83
Zoe	48	0	27

	Core	Bug	Enhancement	Future Dev.	Question	Usability
Web	0:8	1:18	2:5	4:0	0:7	0:9
App	0:17	1:18	7:3	8:0	0:3	2:11
API	0:30	1:33	11:25	12:1	0:8	0:1

Note: x:y means x open issues and y closed issues

Totals:

Web

4 open
50 closed

App

11 open
56 closed

API

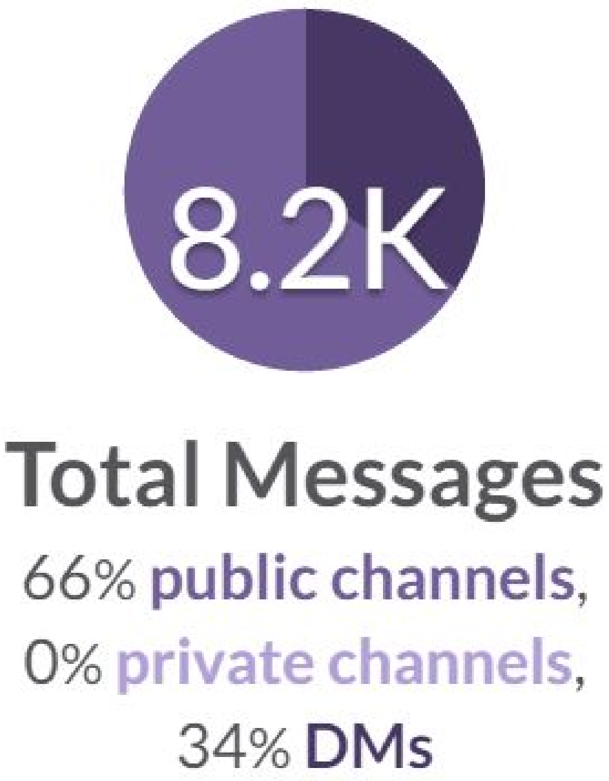
14 open
88 closed

Report/Presentation (this label was created for the final report and presentation milestone)

0 open
14 closed

<u>Total: 228</u>

Slack



Requirements and Limitations

Requirements to run each of the products, and the known limitations of each application

Mobile Application

Requirements

Building/Installing Android from Desktop

JDK8
JavaFX
Gradle
Android SDK 21 & Android Build Tools 21
ANDROID_HOME in gradle.properties file must be set to the path of the top level of the Android SDK directory

Installing Android from Google Play Store

Android OS (version 4.4 'kitkat' or above)

Building/Installing iOS from Desktop

Device with MacOS
JDK8
JavaFX
Gradle
XCode 7 (if a free provisioning profile is used) or
XCode 6 or XCode 7 (if a paid developer profile is used)

Limitations

- iOS does not support audio signals
- As the app relies on GPS for navigation, the performance of the app (in terms of locating traps and accurately mapping traps) is limited by the capabilities of the device (some low-end mobile handsets do not include a GPS unit) and/or the accuracy of GPS which at times can be inaccurate for reasons which are unknown
- When traplines span an area requiring 500 or more map tiles to be displayed, the 'pre-load' function does not work. This is a limitation designed to prevent the mobile app from downloading and storing large numbers of map tiles in the cache
- There is a known bug with map tile loading which occurs when traplines span both sides of the equator (Issue #185 on Github Issue Tracker)

Web Application

Requirements

Apache web server on Amazon EC2 instance
AWS

Limitations

- Administration pages do not scale for mobile

API

Requirements

JDK1.8
Apache2 web server
Tomcat 7+
PostgreSQL Database 9.5
PostgreSQL JDBC driver 9.4.1209
Gradle

Limitations

- API does not support hooks or plugins i.e. the capability to execute arbitrary code on the server as a pre- or post-trigger to specific requests
  (This would be useful for sending emails to new users asking them to confirm their account creation, or would allow session login to be implemented more elegantly using a simple dataset rather than simply overriding the servlet URL listener with a higher-priority servlet to manage login. This could be an avenue worth exploring to make the API more dynamic in the future)

Installation Manual

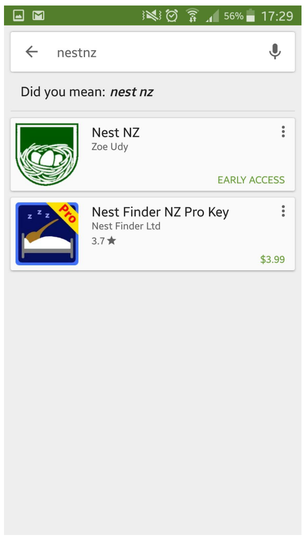
The following sections describe installation for the general public

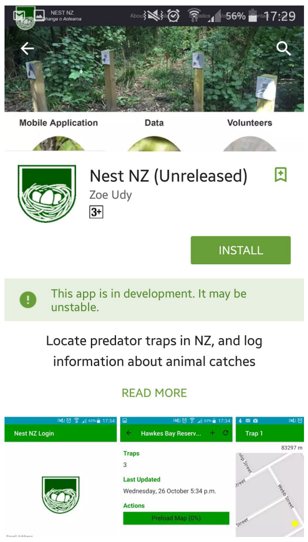
Note: to find information on how to run code in a development environment view 'Guide to Checking Out and Building from the Repository' where the initialisation of the database is outlined

Android

How to install the mobile application on devices running the Android Operating System

1. Open the Play Store application from your mobile device running Android OS version 4.4 or above
2. Search 'nestnz'
3. Click the Nest NZ mobile application icon (shown in the first screenshot below)
4. Click install





Note: To sign in to the mobile application an account will have to be made by an admin user via the web application

iOS

How to install the mobile application on iOS

The mobile application is not currently available on any publicly accessible platforms, however, it can be installed using the steps detailed in the *Guide to Checking Out and Building from the Repository* located below.

Customisation Manual

How to change style and authentication

Mobile Application Customisation

The styles and colours for the mobile app can be changed by modifying the main stylesheet, located in ProjectNest-App/src/main/resources/org/nestnz/app/styles.css

Authentication

Authentication can be customised by changing a user's access level. The access levels follow the permissions model (detailed in Appendix A - RESTful API Specification). Users are also able to log in using either a username or password (distinguished by the request handler by the presence or absence of an '@' character).

Guide to Checking Out and Building from the Repository

Location of the build scripts and other information needed to check out the mobile product from the repository

Release is here: https://github.com/FrancisG-Massey/Capstone2016/releases/tag/v1.1

A zip file containing the repository can be downloaded from here or, alternatively, the 'git' command can be used to check out the repository:

```
git clone https://github.com/FrancisG-Massey/Capstone2016.git
```

Checking out the specific release:

```
git clone -b v1.1 --depth 1
https://github.com/FrancisG-Massey/Capstone2016.git
```

API Database

The ProjectNest-API requires a nestnz PostgreSQL 9.5+ database for all data storage. This can be setup on localhost or externally (AWS RDS etc).

Linux

The following commands can be used to set up postgres on localhost:

```
apt-get install postgresql-9.5
apt-get install pgadmin3            #optional GUI for postgres
```

Windows

Download and follow the install instructions for PostgresSQL/pgadmin:

https://www.postgresql.org/download/
https://www.pgadmin.org/

Post-Installation

The database can be setup in postgres using the following steps, and require a basic understanding of PostgreSQL. These can be done easily from the object browser in pgadmin3.

Follow instructions in the file: ProjectNest-API/db/schema_full.sql:

1. Create a login role 'nestnz' ensuring it has createdb permissions

2. Create a database 'nestnz' ensuring it has default schema 'public'
3. Run the SQL script on the nestnz database to create the database schema.

API Request Handler

Linux

While many modern linux distributions come with Java 8+, Tomcat 7+, and Gradle 1.4+ pre-installed, make sure that you have these set up:

```
    sudo apt-get install openjdk-8-jdk          #install java 8
    sudo apt-get install gradle                 #install gradle
    sudo apt-get install tomcat7                #install tomcat
    $CATALINA_HOME/bin/startup.sh               #start tomcat
```

After checking out the repository, run the commands listed in the following sections to run the server. <git-clone-dir> refers to the directory which you cloned the repository into above:

```
    cd <git-clone-dir>
    gradle war
    cp <git-clone-dir>/ProjectNest-API/build/libs/ProjectNest-API.war
/var/lib/tomcat7/webapps/nestnz.war
```

The ProjectNest-API should now be listening in the tomcat container (default port is 8080). HTTP Requests to the API root (http://localhost:8080/api/) should now connect.

Next we need to point it to the nestnz database. Create a new file to store the database connection properties:

```
    /var/lib/tomcat7/webapps/nestnz/WEB-INF/dbconfig.properties
```

Ensure the file contains these four properties, configured as required to point to your nestnz database:

```
    driver=org.postgresql.Driver
    url=jdbc\:postgresql\://localhost\:5432/nestnz
    user=nestnz
    password=SomeVeryStrongPassword123456789
```

Windows

Download and follow the installation instructions for Java, Tomcat7+, and Gradle:

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
http://tomcat.apache.org/
https://gradle.org/gradle-download/

As above, build the API WAR file with Gradle and copy it into the tomcat webapps folder. With tomcat's default auto-deploy setting enabled, this will automatically deploy it to listen on the tomcat port on localhost.

As above create the dbconfig.properties file in the deployed nestnz API directory. With default install settings, this path may be similar to:

C:\Program Files\Apache Software Foundation\Apache Tomcat 8.0.27\webapps\nestnz\dbconfig.properties

Also as above, ensure the file contains these four properties, configured as required to point to your nestnz database:

```
driver=org.postgresql.Driver
url=jdbc\:postgresql\://localhost\:5432/nestnz
user=nestnz
password=SomeVeryStrongPassword123456789
```

Post-Installation

With the nestnz request handler (and necessarily, the database too) now set up, It should be possible to log into the system via the web UI (if also set up on localhost).

New admin users can be created by logging in with the default admin user using the following credentials:

Username: nestrootadmin
Password: nestrootadmin

After logging in using the web UI, new users (including admins) can be added on the "Users" page, accessible under the "admin" dropdown on the navigation bar at the top of the site.

We recommend disabling the nestrootadmin user login once normal administrator accounts have been set up.

Finally, the service can be set up for public access via apache2 etc, using your domain name and HTTPS certificate. For more details, visit https://httpd.apache.org/

Mobile

After checking out the repository and changing into the "ProjectNest-App" directory, run one of the commands listed in the following sections to build for your desired platform.
Note: You can also replace "gradle" with "./gradlew" (or just "gradlew" on Windows) in the following commands if you don't have Gradle installed on your computer.

Desktop

If you just want to show the app on your computer, run this commands:

```
gradle run
```

This command will download any dependencies needed & open a window with the app running in it.

Android

To install the app on a mobile device running Android, first run this command to build the debug APK:

```
gradle android
```

This command will create a file called "ProjectNest-App.apk" under "build/javafxports/android". You can then copy this file onto your android device via whatever means you prefer (using a USB cable, dropbox, Google Drive, etc), open it, and select "install" (you might have to enable "installing third party apps" before you're able to install it).

iOS

Installing the app on iOS requires a provisioning profile to be created first. To do so, make sure Xcode 7 is installed then create a free provisioning profile for "org.nestnz.app":

1. Create a new project by filling product, organization names, and organization identifier
2. Select your iOS device
3. Navigate to 'No Matching provisioning profile found' and click 'fix issue' button, which will provide a free provisioning profile under your account for 7 days

Note: An Apple id is required to register as a developer in XCode

Once this profile has been created, use one of these commands to build & launch the app:

For a connected device:

```
gradle launchIOSDevice
```

For a simulator, use either:

```
gradle launchIPhoneSimulator
```

or

```
gradle launchIPadSimulator
```

Commercialisation Plan

Future plans for Project Nest

The Nest NZ mobile application is available on the Google Play Store as a 'beta' release application. The link to this product will be distributed to key project stakeholders and Department of Conservation volunteers known to the team members, feedback will then be collected (via email) and any changes suggested reviewed and implemented. Following this process, the mobile application will be publically published on the Google Play Store and deployed to the Apple App Store.

The Project Nest Github repository has now been made public with the view to facilitate and encourage ongoing open source development for this project.

There are several issues that remain open on the Github Issue Tracker - most are tagged with the labels, 'enhancement' or 'future development' meaning that they were always issues that had low priority and were not intended to be completed during the initial development cycle (the twelve weeks of semester).

The open issues (created by team members only) on the issue tracker are listed below:

#294 Update trap_lastresettimestamp on POST/catch

Tags: **enhancement**, **future development**, **API**, **App**

#290 Support logging of damage and hazards to traps and tracks

Tags: **enhancement**, **future development**, **App**, **API**, **Web**

#272 Deleting users/traplines should cascade to enrollments

Tags: **enhancement**, **future development**, **API**

#271 POST/PUT requests to inactive entities should fail

Tags: **enhancement**, **future development**, **API**

#259 Add trapline and user filters for trapline-user UI context

Tags: **enhancement**, **future development**, **API**

#257 'Other' option for catches

Tags: **enhancement**, **App**

#249 Audio signals on iOS

Tags: **enhancement**, **App**

#246 User dataset should contain user's trapline count

Tags: **enhancement**, **future development**, **API**

#220 Create request failure policy

Tags: **enhancement**, **App**

#191 Add admin option to show inactive entities

Tags: **enhancement**, **future development**, **Web**

#185 Map load status - negative tile count

Tags: **bug**, **future development**, **App**

#177 Refactor the API to use a single centralised config file

Tags: **enhancement**, **future development**, **API**

#176 Improve error reporting in API responses

Tags: **enhancement**, **future development**, **API**

#145 Add functionality for taking photos in mobile app

Tags: **enhancement**, **future development**, **App**

#138 Support dataset pre-parsing and caching in memory

Tags: **enhancement**, **future development**, **API**

#134 Sensitivity when pressing buttons [in mobile app]

Tags: **usability**, **future development**, **App**

#115 Modifying a trap record's location should transparently create new

Tags: **enhancement**, **future development**, **API**

#113 Modify catch option

Tags: **usability**, **enhancement**, **future development**, **App**

#92 App logout

Tags: **future development**, **App**

#12 Authentication Failed

Tags: **bug**, **future development**, **Web**, **API**

#129 Support image storage/retrieval with AWS S3 buckets

Tags: **future development**, **API**, **Web**, **App**

Key items for future development, which were highlighted in the product presentation, include supporting damage and hazard logging via the mobile application, and displaying graphs and statistics about the data recorded through Project Nest on a publically accessible web page.

Generating interest in Project Nest through our social media channels will be important in securing the support of the Department of Conservation, and in recruiting a group of developers who are willing to contribute their time to the project. The social media account details follow:
https://www.facebook.com/ProjectNestNZ/
https://twitter.com/NestNZ
https://www.youtube.com/channel/UCHzvm0KavuUERAV-V_D28Bg or https://goo.gl/M8Qvk9

Video Tutorial

Links to video tutorials for each application

Web Application Tutorial

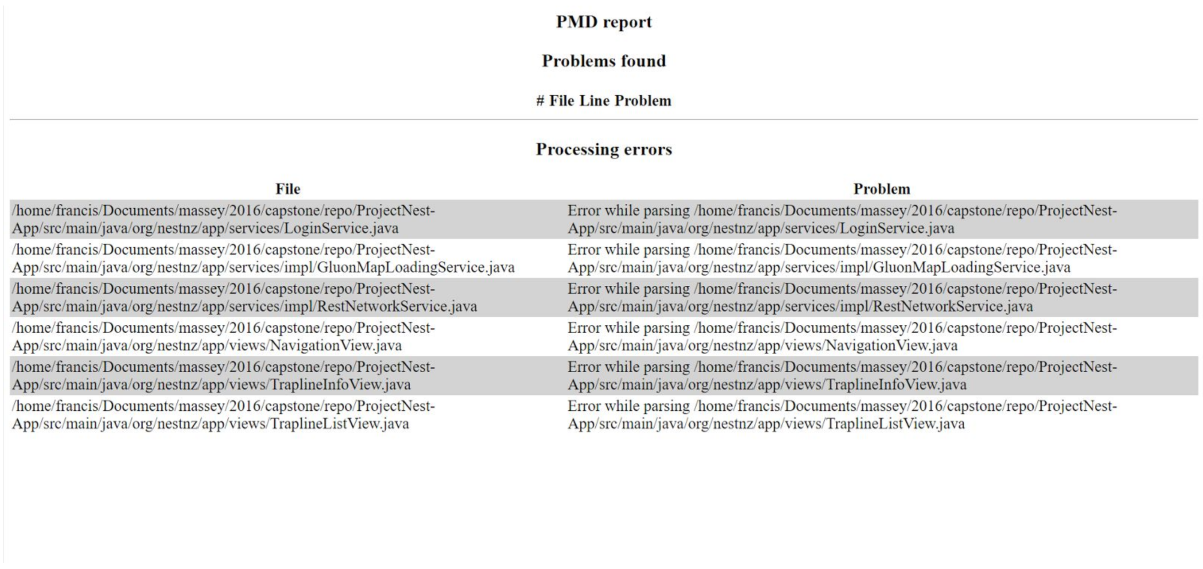
https://www.youtube.com/watch?v=5fU1BA2NKaU
OR
https://goo.gl/LzIKri

Mobile Application Tutorial

Part 1 : https://youtu.be/I0YSNPExt68
Part 2: https://youtu.be/A4INpL_m4DA

Appendix A

PMD Report for Mobile Application



PMD report

Problems found

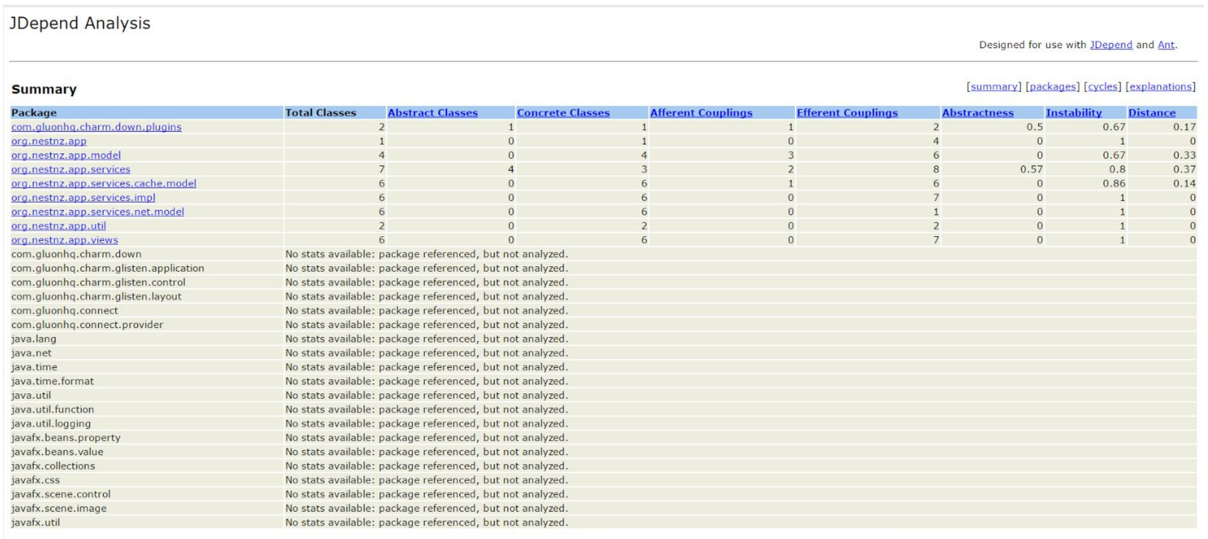
# File Line Problem

Processing errors

File	Problem
/home/francis/Documents/massey/2016/capstone/repo/ProjectNest-App/src/main/java/org/nestnz/app/services/LoginService.java	Error while parsing /home/francis/Documents/massey/2016/capstone/repo/ProjectNest-App/src/main/java/org/nestnz/app/services/LoginService.java
/home/francis/Documents/massey/2016/capstone/repo/ProjectNest-App/src/main/java/org/nestnz/app/services/impl/GluonMapLoadingService.java	Error while parsing /home/francis/Documents/massey/2016/capstone/repo/ProjectNest-App/src/main/java/org/nestnz/app/services/impl/GluonMapLoadingService.java
/home/francis/Documents/massey/2016/capstone/repo/ProjectNest-App/src/main/java/org/nestnz/app/services/impl/RestNetworkService.java	Error while parsing /home/francis/Documents/massey/2016/capstone/repo/ProjectNest-App/src/main/java/org/nestnz/app/services/impl/RestNetworkService.java
/home/francis/Documents/massey/2016/capstone/repo/ProjectNest-App/src/main/java/org/nestnz/app/views/NavigationView.java	Error while parsing /home/francis/Documents/massey/2016/capstone/repo/ProjectNest-App/src/main/java/org/nestnz/app/views/NavigationView.java
/home/francis/Documents/massey/2016/capstone/repo/ProjectNest-App/src/main/java/org/nestnz/app/views/TraplineInfoView.java	Error while parsing /home/francis/Documents/massey/2016/capstone/repo/ProjectNest-App/src/main/java/org/nestnz/app/views/TraplineInfoView.java
/home/francis/Documents/massey/2016/capstone/repo/ProjectNest-App/src/main/java/org/nestnz/app/views/TraplineListView.java	Error while parsing /home/francis/Documents/massey/2016/capstone/repo/ProjectNest-App/src/main/java/org/nestnz/app/views/TraplineListView.java

Appendix B

JDepend Analysis for Mobile Application



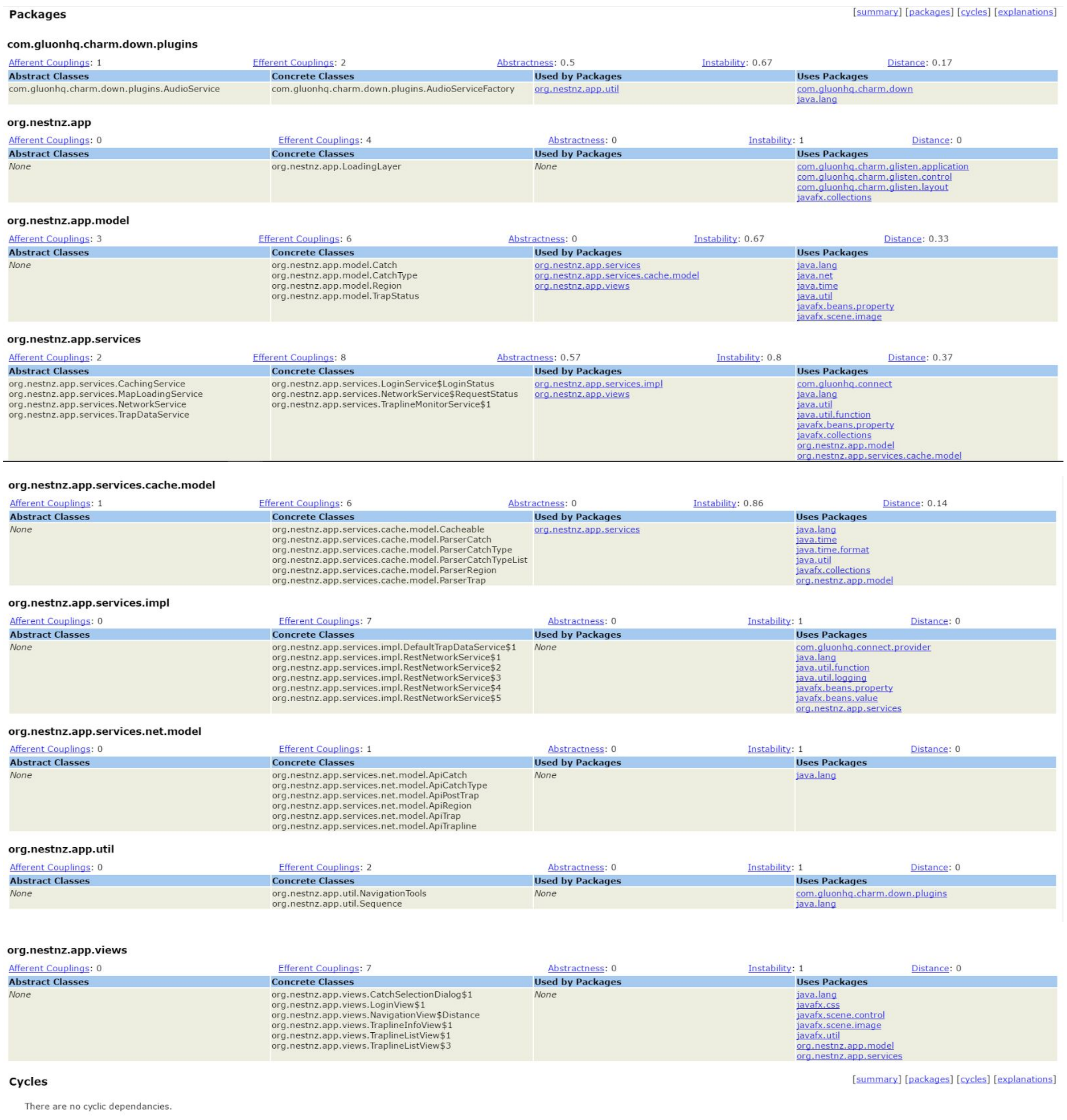
JDepend Analysis

Designed for use with JDepend and Ant.

Summary

[summary] [packages] [cycles] [explanations]

Package	Total Classes	Abstract Classes	Concrete Classes	Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance
com.gluonhq.charm.down.plugins	2	1	1	1	2	0.5	0.67	0.17
org.nestnz.app	1	0	1	0	4	0	1	0
org.nestnz.app.model	4	0	4	3	6	0	0.67	0.33
org.nestnz.app.services	7	4	3	2	8	0.57	0.8	0.37
org.nestnz.app.services.cache.model	6	0	6	1	6	0	0.86	0.14
org.nestnz.app.services.impl	6	0	6	0	7	0	1	0
org.nestnz.app.services.net.model	6	0	6	0	1	0	1	0
org.nestnz.app.util	2	0	2	0	2	0	1	0
org.nestnz.app.views	6	0	6	0	7	0	1	0
com.gluonhq.charm.down	No stats available: package referenced, but not analyzed.							
com.gluonhq.charm.glisten.application	No stats available: package referenced, but not analyzed.							
com.gluonhq.charm.glisten.control	No stats available: package referenced, but not analyzed.							
com.gluonhq.charm.glisten.layout	No stats available: package referenced, but not analyzed.							
com.gluonhq.connect	No stats available: package referenced, but not analyzed.							
com.gluonhq.connect.provider	No stats available: package referenced, but not analyzed.							
java.lang	No stats available: package referenced, but not analyzed.							
java.net	No stats available: package referenced, but not analyzed.							
java.time	No stats available: package referenced, but not analyzed.							
java.time.format	No stats available: package referenced, but not analyzed.							
java.util	No stats available: package referenced, but not analyzed.							
java.util.function	No stats available: package referenced, but not analyzed.							
java.util.logging	No stats available: package referenced, but not analyzed.							
javafx.beans.property	No stats available: package referenced, but not analyzed.							
javafx.beans.value	No stats available: package referenced, but not analyzed.							
javafx.collections	No stats available: package referenced, but not analyzed.							
javafx.css	No stats available: package referenced, but not analyzed.							
javafx.scene.control	No stats available: package referenced, but not analyzed.							
javafx.scene.image	No stats available: package referenced, but not analyzed.							
javafx.util	No stats available: package referenced, but not analyzed.							



Packages

[summary] [packages] [cycles] [explanations]

com.gluonhq.charm.down.plugins

Afferent Couplings: 1 Efferent Couplings: 2 Abstractness: 0.5 Instability: 0.67 Distance: 0.17

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
com.gluonhq.charm.down.plugins.AudioService	com.gluonhq.charm.down.plugins.AudioServiceFactory	org.nestnz.app.util	com.gluonhq.charm.down java.lang

org.nestnz.app

Afferent Couplings: 0 Efferent Couplings: 4 Abstractness: 0 Instability: 1 Distance: 0

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	org.nestnz.app.LoadingLayer	*None*	com.gluonhq.charm.glisten.application com.gluonhq.charm.glisten.control com.gluonhq.charm.glisten.layout javafx.collections

org.nestnz.app.model

Afferent Couplings: 3 Efferent Couplings: 6 Abstractness: 0 Instability: 0.67 Distance: 0.33

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	org.nestnz.app.model.Catch org.nestnz.app.model.CatchType org.nestnz.app.model.Region org.nestnz.app.model.TrapStatus	org.nestnz.app.services org.nestnz.app.services.cache.model org.nestnz.app.views	java.lang java.net java.time java.util javafx.beans.property javafx.scene.image

org.nestnz.app.services

Afferent Couplings: 2 Efferent Couplings: 8 Abstractness: 0.57 Instability: 0.8 Distance: 0.37

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
org.nestnz.app.services.CachingService org.nestnz.app.services.MapLoadingService org.nestnz.app.services.NetworkService org.nestnz.app.services.TrapDataService	org.nestnz.app.services.LoginService$LoginStatus org.nestnz.app.services.NetworkService$RequestStatus org.nestnz.app.services.TraplineMonitorService$1	org.nestnz.app.services.impl org.nestnz.app.views	com.gluonhq.connect java.lang java.util java.util.function javafx.beans.property javafx.collections org.nestnz.app.model org.nestnz.app.services.cache.model

org.nestnz.app.services.cache.model

Afferent Couplings: 1 Efferent Couplings: 6 Abstractness: 0 Instability: 0.86 Distance: 0.14

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	org.nestnz.app.services.cache.model.Cacheable org.nestnz.app.services.cache.model.ParserCatch org.nestnz.app.services.cache.model.ParserCatchType org.nestnz.app.services.cache.model.ParserCatchTypeList org.nestnz.app.services.cache.model.ParserRegion org.nestnz.app.services.cache.model.ParserTrap	org.nestnz.app.services	java.lang java.time java.time.format java.util javafx.collections org.nestnz.app.model

org.nestnz.app.services.impl

Afferent Couplings: 0 Efferent Couplings: 7 Abstractness: 0 Instability: 1 Distance: 0

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	org.nestnz.app.services.impl.DefaultTrapDataService$1 org.nestnz.app.services.impl.RestNetworkService$1 org.nestnz.app.services.impl.RestNetworkService$2 org.nestnz.app.services.impl.RestNetworkService$3 org.nestnz.app.services.impl.RestNetworkService$4 org.nestnz.app.services.impl.RestNetworkService$5	*None*	com.gluonhq.connect.provider java.lang java.util.function java.util.logging javafx.beans.property javafx.beans.value org.nestnz.app.services

org.nestnz.app.services.net.model

Afferent Couplings: 0 Efferent Couplings: 1 Abstractness: 0 Instability: 1 Distance: 0

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	org.nestnz.app.services.net.model.ApiCatch org.nestnz.app.services.net.model.ApiCatchType org.nestnz.app.services.net.model.ApiPostTrap org.nestnz.app.services.net.model.ApiRegion org.nestnz.app.services.net.model.ApiTrap org.nestnz.app.services.net.model.ApiTrapline	*None*	java.lang

org.nestnz.app.util

Afferent Couplings: 0 Efferent Couplings: 2 Abstractness: 0 Instability: 1 Distance: 0

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	org.nestnz.app.util.NavigationTools org.nestnz.app.util.Sequence	*None*	com.gluonhq.charm.down.plugins java.lang

org.nestnz.app.views

Afferent Couplings: 0 Efferent Couplings: 7 Abstractness: 0 Instability: 1 Distance: 0

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	org.nestnz.app.views.CatchSelectionDialog$1 org.nestnz.app.views.LoginView$1 org.nestnz.app.views.NavigationView$Distance org.nestnz.app.views.TraplineInfoView$1 org.nestnz.app.views.TraplineListView$1 org.nestnz.app.views.TraplineListView$3	*None*	java.lang javafx.css javafx.scene.control javafx.scene.image javafx.util org.nestnz.app.model org.nestnz.app.services

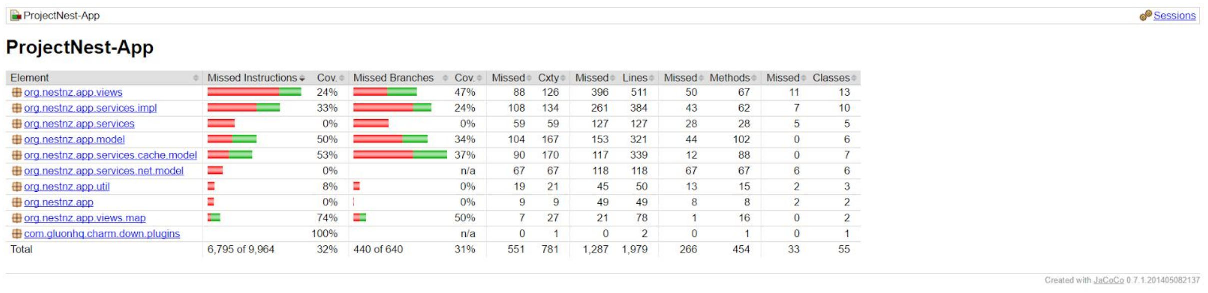
Cycles

[summary] [packages] [cycles] [explanations]

There are no cyclic dependancies.

Appendix C

JaCoCo Report



ProjectNest-App | Sessions

ProjectNest-App

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
org.nestnz.app.views	🟥🟩	24%	🟥🟩	47%	88	126	396	511	50	67	11	13
org.nestnz.app.services.impl	🟥🟩	33%	🟥🟩	24%	108	134	261	384	43	62	7	10
org.nestnz.app.services	🟥	0%	🟥	0%	59	59	127	127	28	28	5	5
org.nestnz.app.model	🟥🟩	50%	🟥🟩	34%	104	167	153	321	44	102	0	6
org.nestnz.app.services.cache.model	🟥🟩	53%	🟥🟩	37%	90	170	117	339	12	88	0	7
org.nestnz.app.services.net.model	🟥	0%		n/a	67	67	118	118	67	67	6	6
org.nestnz.app.util	🟥	8%	🟥	0%	19	21	45	50	13	15	2	3
org.nestnz.app	🟥	0%	🟥	0%	9	9	49	49	8	8	2	2
org.nestnz.app.views.map	🟥🟩	74%	🟥🟩	50%	7	27	21	78	1	16	0	2
com.gluonhq.charm.down.plugins		100%		n/a	0	1	0	2	0	1	0	1
Total	6,795 of 9,964	32%	440 of 640	31%	551	781	1,287	1,979	266	454	33	55

Created with JaCoCo 0.7.1.201405082137

Appendix D

NESTNZ Website Test Cases

Public Pages Test Cases

URL	Description	Asserts
Index	Index page contents	● A navbar element is present at the top of the page ● NESTNZ logo is present on the navbar ● NESTNZ text is present on the navbar ● ‘About’ link element is present ● ‘About’ link text ● ‘Mobile App’ link element is present ● ‘Mobile App’ link text ● ‘Statistics’ link element is present ● ‘Statistics’ link text ● ‘Volunteer link’ element is present ● ‘Volunteer link’ text ● ‘Contact link’ element is present ● ‘Contact link’ text ● ‘Login link’ element is present ● ‘Login link’ text ● Main Header image is present
/#/home	Home page contents	● Image element(Mobile application) 1 is present ● Image 1 title text ● Image 1 description

		● Image element(Data) 2 is present ● Image 2 title text ● Image 2 description ● Image element(Volunteers) 3 is present ● Image 3 title text ● Image 3 description
/#/About	About page contents	● “Contact” title is present ● Image 1 element(Contribute) is present ● “Contribute” title is present ● “Contribtue” description is present ● Image 2 element(Connect) is present ● “Connect” title is present ● “Connect” description is present ● Image 3 element(Collaboration) is present ● “Collaboration” title is present ● “Collaboration” description is present
/#/Mobile App	Mobile Page contents	● “Mobile Application” title is present ● “Mobile Application” text element is present ● Image 1 element(“Trapline Selection” is present ● “Trapline Selection” title is present ● “Trapline Selection” description is present ● Image 2 element(“Map”) is present ● “Map” title is present ● “Map” description is present ● Image 3 element(“Select Trapline”) is present ● “Select Trapline” title is present ● “Select Trapline” description is present ● Image 4 element(“Log Catch”) is present ● “Log Catch” title is present ● “Log Catch” description is present ● “Download” title is present ● “Download” description is present ● “Download for google store” button element is present ● “Download for google store” text is present ● “Download for apple store” text is present
/#/Statistics	Statistics Page contents	● “Statistics” title text is present ● “Coming Soon” image element is present ● Image referencing text is present ● Two link elements in the image referencing are present ● “For now…” text is present ● “Export annual report” button element is present ● “Annual report” button text is present
/#/Volunteer	Volunteer Page	● “Volunteer” title text is present

	contents	● "Volunteer" title element is present ● "Volunteering with DOC is a rewarding experience" is present ● "You are contributing to the health of New Zealand's environment" is present ● "If you would like to get involved, apply through your local DOC office by following the link below" is present ● "Apply via DOC" button element is present ● "Apply via DOC" button text is present
/#/Contact	Contact Page contents	● "Contact" title text is present ● "Contact" title element is present ● "Address" image element is present ● "Address" image description is present ● "Email" image element is present ● "Email" image description is present ● "Github repo" image element is present ● "Github repo" image description is present ● Image referencing text is present ● "Freepik" link element is present ● "Pixel Buddha" link element is present ● "www.flaticon.com" link element is present
/#/Login	Login Page contents	● Background image element is present ● "Admin Log In" title text is present ● "Admin Log In" title element is present ● Login form element is present ● Email text is present ● Key icon is present ● Email text box element is present ● Password text is present ● Lock icon is present ● Password text box element is present ● Log In button element is present ● Log In button text is present
/#/Login	Try login with a valid email and password	● Type user email ● Type user password ● Click Login button

Private (Admin) Page Test Cases

URL	Description	Asserts
/#/trapline-admin	Admin menu options will be displayed once	● Type user email ● Type user password ● Click Login button

	a user login to NestNZ web app	• Check Admin link element is appeared on the navigation bar • Click Admin link • Check Traplines link element is present under the Admin link • Check Users link element is present under the Admin link
/#/trapline-admi n	Trapline Page contents	• Type user email • Type user password • Click Login button • Trapline Admin title text is present • Add a trapline link element is present • Check trapline table ○ Check header elements are presents ■ Trapline Name ■ Start ■ End ■ Traps ■ Volunteers ■ Catch History ■ Action Check also above texts are presents • Traps View link element is present • Volunteers View link element is present • Catch History View link element is present • Edit link element is present • Home link text is present on the Breadcrumb • Admin text is present on the Breadcrumb
/#/trapline-admi n	Click View Catch History and check contents in the dialog	• Type user email • Type user password • Click Login button • Click View Catch History link • Check Catch History title element is present • Check the region title element is present • Check the trapline name element is present • Check catch history table ○ Check header elements are presents ■ Trap ■ Catch ■ Date Check also above texts are presents • Check close button element is present • Check close button text is present • Check export button element is present • Check export button text is present • Click export button to download csv file • Close the dialog by clicking ‘x’ button

/#/trap-admin/<trapline-id><trapline name>/	Check trap page contents	<ul><li>Type user email</li><li>Type user password</li><li>Click Login button</li><li>Click View Traps link in trapline page</li><li>Check Trap Admin title text is present</li><li>Check View Map button element is present</li><li>Check View Map button text is present</li><li>Add a trap link is present</li><li>Check trap table<ul><li>Check header elements are presents<ul><li>Trap Number</li><li>Location</li><li>Status</li><li>Last Reset</li><li>Placed Date</li><li>Catch History</li><li>Action</li></ul>Check also above texts are presents</li></ul></li><li>Check Catch history view link element is present</li><li>Check Catch history view text is present</li><li>Check Edit link element is present</li><li>Check Edit link text is present</li><li>Check page navigation number “1” is present</li><li>Check “Prev” text is present</li><li>Check “Next” text is present</li><li>Home link text is present on the Breadcrumb</li><li>Admin text is present on the Breadcrumb</li><li>Trapline name text element is present on the Breadcrumb</li></ul>
/#/trap-admin/<trapline-id><trapline name>/	View traps in a Map	<ul><li>Type user email</li><li>Type user password</li><li>Click Login button</li><li>Click View Traps link in trapline page</li><li>Click View Map button</li><li>Check Trapline name title text is present</li><li>Check map element is present</li><li>Check close button element is present</li><li>Check clost button text is present</li><li>Close the dialog by clicking ‘close’ button</li></ul>
/#/volunteer-admin/<trapline-id><trapline name>/	Check Volunteer admin page contents	<ul><li>Type user email</li><li>Type user password</li><li>Click Login button</li><li>Click View Volunteers link in trapline page</li><li>Check Volunteer Admin title text is present</li><li>Check volunteers table<ul><li>Check header elements are presents<ul><li>ID</li><li>Full Name</li></ul></li></ul></li></ul>

		■ Email ■ Start Date ■ Role Check also above texts are presents ● Check page navigation number “1” is present ● Check “Prev” link is present ● Check “Next” link is present ● Home link element is present on the Breadcrumb ● Admin link element is present on the Breadcrumb ● Trapline name text element is present on the Breadcrumb
/#/trapline-admin/add-newtrapline	Check add new trapline page contents	● Type user email ● Type user password ● Click Login button ● Click Add Trapline link in trapline page ● Check New Trapline title text is present ● Check form element ○ Check input elements are present ■ Name ■ Region ■ Start ■ End ■ Default Bait Type ■ Default Trap Type Check also above texts are presents ● Check page navigation number “1” is present ● Check “Prev” link is present ● Check Save button element is present ● Check Save button text is present ● Home link element is present on the Breadcrumb ● Admin link element is present on the Breadcrumb ● Add new trapline text element is present on the Breadcrumb
/#/trapline-admin/<trapline-id>/edit-trapline		● Type user email ● Type user password ● Click Login button ● Click Edit Trapline link in trapline page ● Check Edit Trapline title text is present ● Check form element ○ Check input elements are present ■ Name ■ Region ■ Start

		■ End ■ Default Bait Type ■ Default Trap Type Check also above texts are presents ● Check input values are present ○ Check name is equal to “Manawatu Gorge Tawa Loop” ○ Check start value is “Ashhurst” ○ Check end value is “Woodville” ● Check Save button element is present ● Check Save button text is present ● Home link element is present on the Breadcrumb ● Admin link element is present on the Breadcrumb ● Add new trapline text element is present on the Breadcrumb
/#/trap-admin/<trapline-id><trapline name>/add-new	Check add new trap page contents	● Type user email ● Type user password ● Click Login button ● Click View Traps link in trapline page ● Click Add Trap link in trap page ● Check New Trap title text is present ● Check form element ○ Check input elements are present ■ Latitude ■ Longitude ■ Bait type ■ Trap type ■ Trap Number ■ Active ■ Inactive Check also above texts are presents ● Check Save button element is present ● Check Save button text is present ● Home link element is present on the Breadcrumb ● Admin link element is present on the Breadcrumb ● Trapline name text element is present on the Breadcrumb ● Add new trap text element is present on the Breadcrumb
/#/trap-admin/<trapline-id><trapline name>/edit-trap	Check edit trap page contents	● Type user email ● Type user password ● Click Login button ● Click View Trap link in trapline page ● Click Edit Trap link in trap page

		● Check Edit Trap title text is present ● Check form element ○ Check input elements are present ■ Latitude ■ Longitude ■ Bait type ■ Trap type ■ Trap Number ■ Active ■ Inactive Check also above texts are presents ● Check input values are present ○ Check Latitude value is equal to ”-40.309037” ○ Check Longitude value is “175.772448” ○ Check end value is “Woodville” ● Check Save button element is present ● Check Save button text is present ● Home link element is present on the Breadcrumb ● Admin link element is present on the Breadcrumb ● Trapline name link element is present on the Breadcrumb ● Trap number text element is present on the Breadcrumb
/#/user-admin	Check user admin page contents	● Type user email ● Type user password ● Click Login button ● Click admin menu ● Click users link ● Check Users title text is present ● Check Add a User link is present ● Check Users table ○ Check header elements are presents ■ Full Name ■ Email ■ Start Date ■ Permission ■ Trapline ■ Action Check also above texts are presents ● Check View Traplines link element is present ● Check View Traplines link text is present ● Check Edit link element is present ● Check Edit link text is present ● Home link text is present on the Breadcrumb ● Home link element is present on the

		Breadcrumb ● Admin text is present on the Breadcrumb
/#/user-admin/add-user	Check add user page contents	● Type user email ● Type user password ● Click Login button ● Click admin menu ● Click users link ● Click Add a User link ● Check New User title text is present ● Check form element ○ Check input elements are present ■ User Email Address ■ User Full Name ■ User Phone ■ User Permission ■ User Password ■ Confirm User Password Check also above texts are presents ● Check Save button element is present ● Check Edit link text is present ● Add new user text is present on the Breadcrumb ● Home text is present on the Breadcrumb ● Home link element is present on the Breadcrumb ● Admin text is present on the Breadcrumb ● Admin link element is present on the Breadcrumb
/#/user-admin/<user-id>edit-user/	Check edit user page contents	● Type user email ● Type user password ● Click Login button ● Click admin menu ● Click users link ● Click Edit button ● Check Edit User:<name> title text is present ● Check form element ○ Check input elements are present ■ User Email Address ■ User Full Name ■ User Phone ■ User Permission ■ User Password ■ Confirm User Password Check also above texts are presents ● Check input values are present ○ User Email Address value is “admin@nestnz.org” ○ User Full Name value is “admin” ○ User Phone value is “0211824040”

		<ul><li>Check Save button element is present</li><li>Check Save button text is present</li><li>Edit &lt;user email> text is present on the Breadcrumb (“admin@nestnz.org”)</li><li>Home text is present on the Breadcrumb</li><li>Home link element is present on the Breadcrumb</li><li>Admin text is present on the Breadcrumb</li><li>Admin link element is present on the Breadcrumb</li></ul>
/#/user-admin/	Check View Traplines dialog and its contents	<ul><li>Type user email</li><li>Type user password</li><li>Click Login button</li><li>Click admin menu</li><li>Click users link</li><li>Check View / Add Traplines for <user email> text is present (“admin@nestnz.org”)</li><li>Check View / Add Traplines subtitle text is present</li><li>Check Registered Traplines caption text is present</li><li>Check view user’s traplines table<ul><li>Check header elements are presents<ul><li>Trapline</li><li>Start</li><li>End</li><li>Role</li></ul></li><li>Check also above texts are presents</li></ul></li><li>Check close button element is present</li><li>Check clost button text is present</li><li>Check add button element is present</li><li>Check add button text is present</li><li>Close the dialog by clicking ‘x’ button</li></ul>

Appendix E

RESTful API Specification

1.0 General API usage

API calls to the Project Nest server use HTTP 1.1 requests and standard REST verbs to map to HTTP methods as outlined below. By default, data is accepted as a single JSON object in the request body, and if returned, is as a JSON array in the response body.

The NestNZ REST API attempts to comply with the RFC 7231 standard for HTTP methods.

Users must log in to the API before they can make API request calls which retrieve or manipulate data entities.

All requests to the API should be encrypted with HTTPS as a basic and necessary security precaution.

Logging in requires a POST request to /session to create a temporary session. Users must supply their username and password in an HTTP Basic Authentication header over HTTPS.

The server will respond with an HTTP 201 Created, or an HTTP 403 Forbidden response depending on whether the user is known to the server.

If the login attempt is successful, the server will issue a session token in a custom HTTP header named exactly: “Session-Token”. This session token header and exact value should be added to each subsequent API call to the server until they wish to logout.

Session tokens will expire after 30 minutes of inactivity, so interfacing clients should be prepared to reconnect in the case of a timeout (403 returned).

1.1 API base URL

API Requests should be made to the following base URL:

https://api.nestnz.org/

The API currently does not have CORS enabled, thus has been made temporarily available to the webapp via the following base URL:

https://www.nestnz.org/api/

All API requests should be made as a subroute of the above URL.
Supported subroutes are listed below.

1.2 Supported entity paths

- `/user`
- `/session`
- `/region`
- `/trapline`
- `/trap`
- `/catch`
- `/catch-type`
- `/bait`
- `/trapline-user`

1.3 General API request structure

In many cases, access to entities will follow this general structure:

Generic Operation	**REST Verb**	**Generic Resource URL**
Fetch entity record(s) in table	GET	/<entity> /<entity>/<id> /<entity>[?filter1=val1][&filter2=val2]
Add a new entity to the table	POST	/<entity>
Update an entity's values	PUT	/<entity >/<id>
Remove an entity from the table or flag inactive	DELETE	/<entity >/<id>

POST requests which include an entity id are syntactically incorrect and will fail with 400 bad request, as will PUT or DELETE requests which do not include an entity id.

In some cases, columns may be hidden or read-only due to the nature of the data.
E.g.

- A hashed user password
- A timestamp of creation when a catch was logged

1.4 List of all supported API calls

Following is a comprehensive list of all API calls supported by the Project Nest server. Common calls intended for use with the mobile application are highlighted in blue while others are intended for both the mobile application and for administrative purposes within the web application.

Operation	**REST Verb**	**Resource URL**
Create session	POST	/session
Logout	DELETE	/session
Create a new trapline	POST	/trapline
Retrieve trapline(s)	GET	/trapline[?<params>]
Update a trapline	PUT	/trapline/<id>
Remove a trapline	DELETE	/trapline/<id>
Create a new trap	POST	/trap
Retrieve trap(s)	GET	/trap[?<params>]
Update a trap	PUT	/trap/<id>
Remove a trap	DELETE	/trap/<id>
Log a catch from a trap	POST	/catch
Retrieve catch record(s)	GET	/catch[?<params>]
Add a new type of trap	POST	/trap-type
Retrieve trap type(s)	GET	/trap-type[?<params>]
Update a trap type	PUT	/trap-type/<id>
Remove a trap type	DELETE	/trap-type/<id>
Add a new region	POST	/region
Retrieve region(s)	GET	/region[?<params>]
Update a region	PUT	/region/<id>
Remove a region	DELETE	/region/<id>
Add a new user	POST	/user

Retrieve user data	GET	/user[?<params>]
Update a user	PUT	/user/<id>
Remove a user	DELETE	/user/<id>
Add a new role to a trapline for a user	POST	/trapline-user
Retrieve user role data	GET	/trapline-user[?<params>]
Update user role	PUT	/trapline-user/<id>
Delete user role	DELETE	/trapline-user/<id>
Add a new type of catch	POST	/catch-type
Retrieve types of catches	GET	/catch-type[?<params>]
Update catch type data	PUT	/catch-type/<id>
Remove catch type	DELETE	/catch-type/>id>
Add new bait type	POST	/bait
Get bait types	GET	/bait[?<params>]
Update bait type	PUT	/bait/<id>
Remove bait type	DELETE	/bait/<id>

1.5 Permissions for API access

1.5.0 Access roles

User admin

User-admins can be considered to have system-wide administrator access to all entities, and can generally read/write to all of these which are readable/writable. User-admins have an “admin” flag on their user entity.

Trapline Admin

Trapline-admins can be considered to have administrator access to the trapline to which they are assigned to, and can freely edit this trapline or its traps, and create users or enrol users to it. Trapline-admins have an “admin” flag on their trapline-user entity.

Non-admin

Users without either a user-admin true flag or a trapline-admin true flag on any trapline are considered to be non-admins. Non-admins can only access regions/traplines/traps/catches/enrolments or users which are enrolled on a trapline which they themselves are enrolled on.

Non-user

Users who make requests to the API without first authenticating or logging in are considered to be non-users. Non-users can generally not access any of the defined data entities, and may only a few select from a few exposed catch-reports or summaries which are publicly available.

1.5.1 Access permissions by role

Action (down) Role (across)	User Admin	Trapline Admin	Non-Admin
Create sessions (Login)	Self only	Self only	Self only
View sessions	Yes	No	No
Edit sessions	No	No	No
Delete sessions (Logout)	Self only	Self only	Self only

Create users (Incl. grant admin)	Yes	Can create users, can't grant admin	No
View users	All	Self, Other users in same enrolled traplines	Self, Other users in same enrolled traplines
Edit users (Incl. grant admin)	Yes	Self only	Self only
Delete users	Yes	Self only	Self only
Create regions	Yes	No	No
View regions	Yes	Regions containing enrolled traplines only	Regions containing enrolled traplines only
Edit regions	Yes	No	No
Delete regions	Yes (empty only)	No	No
Create traplines	Yes	No	No
View traplines	Yes	Enrolled traplines only	Enrolled traplines only
Edit traplines	Yes	In enrolled traplines only	No
Delete traplines	Yes (empty only)	In enrolled traplines only	No
Create traps	Yes	In enrolled traplines only	No
View traps	Yes	In enrolled traplines only	In enrolled traplines only
Edit traps	Yes	In enrolled traplines only	No
Delete traps	Yes	In enrolled traplines only	No
Create catches	No	In enrolled traplines only	In enrolled traplines only
View catches	Yes	In enrolled traplines only	In enrolled traplines only
Edit catches	No	No	No

Delete catches	No	No	No
Create trapline-users (Incl. grant trapline admin)	Yes	In admin-enrolled traplines only	No
View trapline-users	Yes	In enrolled traplines only	In enrolled traplines only
Edit trapline-users (grant/revoke trapline admin)	Yes	In enrolled traplines only	No
Delete trapline-users, Disenroll from trapline	Yes	In enrolled traplines only	Self only
Create bait	Yes	Yes	No
View bait	Yes	Yes	Yes
Edit bait	Yes	Yes	No
Delete bait	Yes (unused only)	Yes (unused only)	No
Create trap-type	Yes	Yes	No
View trap-type	Yes	Yes	Yes
Edit trap-type	Yes	Yes	No
Delete trap-type	Yes (unused only)	Yes (unused only)	No
Create catch-type	Yes	Yes	No
View catch-type	Yes	Yes	Yes
Edit catch-type	Yes	Yes	No
Delete catch-type	Yes (unused only)	Yes (unused only)	No

1.6 List of API HTTP headers

Field	**Value / Description**
“Accept” header	“application/json”, “application/*”, “*/json” or “*/*” String. All objects and messages returned by the API will be encoded in JSON for portability. Include this in each GET request which expects a JSON response.
“Authorization” header	“Basic <base64credentials>” Credentials should be a string of the form “username:password” and should be encoded in base64 to allow non-alphabetic/numeric symbols. Include this in each POST request to /session ONLY.
“Content-Type” header	“application/json” String. All objects sent to the API for processing should be encoded in JSON for size and portability. Include this in each POST and PUT request with a JSON body.
“Content-Length” header	“<number>” The size in bytes of the response body. Returned by GET requests.
“Session-Token” header	“<session_token>” This is a secure UUID-formatted string returned as a header by the API on login, and on subsequent requests while logged in. Include this in EVERY (non-login) request.
“Date” header	“Sun, 30 Oct 2016 11:59:29 GMT” RFC 2616 formatted timestamp, specifying the time that the server response was generated.
“Expiry” header	“Sun, 30 Oct 2016 12:29:29 GMT” RFC 2616 formatted timestamp, specifying the session expiry, updated by each request made while logged in.
“Access-Control-Allow-Origin”	“www.nestnz.org” CORS Header to explicitly enable XHR access from the webapp. Reserved for future use.
“Access-Control-Allow-Methods”	“GET,POST,PUT,DELETE” CORS Header to explicitly enable XHR access from the webapp. Reserved for future use.
“Access-Control-Allow-Headers”	“Origin, X-Requested-With, Content-Type, Accept” CORS Header to explicitly enable XHR access from the webapp. Reserved for future use.

1.7 List of API HTTP response codes

Many HTTP response codes are used generically across the API:

200 OK	Successful GET request. Content in response body.
201 Created	Successful login request. Token in response header. Successful POST request. Location to new resource in header.
204 No Content	Successful GET request with no return entities. Successful PUT request, entity has been updated. Successful DELETE request, entity inactive/removed.
400 Bad Request	Missing headers or poor request/header syntax, etc. Entity conflict, invalid permissions
401 Unauthorized	Missing or poorly formed Basic Auth header during login attempt.
403 Forbidden	Valid request but not logged in or session is timed out.
404 Not Found	Unrecognised entity path.
405 Method Not Allowed	A default response to an undocumented and unsupported HTTP method request.
409 Conflict	Reserved for future use.
500 Internal Server Error	System/programming bug in the API. Please contact support.
501 Not Implemented	Requested method/functionality will be implemented soon.

2.0 Comprehensive list of entities

2.1 Session Entities

2.1.0 Summary

Session entities represent a logged-in state of a user. The behave differently from most other entities represented in the API. A user will login with their credentials to create one and receive a session token, use the token each time they make other API requests, and then use the token again to logout when they are done, invalidating the token.

2.1.1 Creating session entities

New session entities can be created by making a POST request to the API URL (see introduction), appended with the '/session' subroute.

An HTTP Basic Authentication header is required for this request. I.e.:

```
POST /session HTTP/1.1
HOST: api.nestnz.org
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ
```

A successful response will have no body, but will contain a secure UUID-formatted "Session-Token" header:

```
HTTP/1.1 201 Created
Date: Wed, 21 Sep 2016 12:19:56 GMT
Server: Apache-Coyote/1.1
Session-Token: 336b99f7-732e-4739-8af9-f1661e065dce
Connection: keep-alive
```

The client can consider themselves to be logged in upon receiving this token, and should copy this header to subsequent API requests.

2.1.2 Deleting session entities

Session entities can be deleted by making a DELETE request to the API URL (see introduction), appended with the '/session' subroute. For example:

```
DELETE /session HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

A successful response will have no body, but may be determined to be successful by the HTTP status code.

```
HTTP/1.1 204 No Content
Date: Wed, 21 Sep 2016 12:19:56 GMT
Server: Apache-Coyote/1.1
Connection: keep-alive
```

Making a DELETE request on a session entity will log the user out from that session permanently. Once the request has been processed, the session will be invalid, and the user will need to create a new session before further API requests will be accepted.

2.2 User Entities

2.2.0 Summary

User entities represent a user. The contain information about the user’s account, their login credentials, contact details and other information. The entity stores whether the user is an admin or not, and also whether their account is enabled, and these are used to determine the user’s permissions when making data requests.

2.2.2 Retrieving user entities

User objects can be retrieved from the API URL (see introduction) appended with ‘/user’.
For example:

```
GET /user HTTP/1.1
Host: api.nestnz.org
Session-Token: 336b99f7-732e-4739-8af9-f1661e065dce
Accept: application/json
```

An array of retrieved user objects will be returned on success, following the basic form:

```
HTTP/1.1 200 OK
Date: Wed, 21 Sep 2016 12:19:56 GMT
Server: Apache-Coyote/1.1
```

```
Content-Type: application/json
Content-Length: 135
Connection: keep-alive
[{
    "id": 2,
    "username": "shuntdev",
    "fullname": "shunt, developer",
    "email": "shunt@nestnz.org",
    "phone": "0123456789",
    "created_timestamp": "2016-09-20 02:16:37.739242",
    "created_user_id": 1,
    "admin": true,
    "inactive": false,
    "can_edit": true
}]
```

By adding an id subroute to the request URL, a single user entity may be retrieved, i.e.
`GET /user/42`

By adding query parameters to a request URL, all entities (that the user has permission to access) which match the parameters may be retrieved:
`GET /user?id=42`

Available filters include:

- id
- created-user-id
- name
- admin
- Inactive (only available to admins)

Filters that may be added later include:

- Region-id
- Trapline-id
- Session-id
- Catch-id

2.2.3 Creating user entities

New user entities can be created by making a POST request to the API URL (see introduction), appended with the ‘/user’ subroute. For example:

```
POST /user HTTP/1.1
HOST: api.nestnz.org
Session-Token: 336b99f7-732e-4739-8af9-f1661e065dce

{
    "username": "new_username",
    "password": "some_strong_password",
    "fullname": "new user",
    "phone": "0123456789",
    "email": "newuser@example.com",
    "admin": false
}
```

A successful response will have no body, but will contain a location header specifying an identifier with which the created resource can be retrieved:

```
HTTP/1.1 201 Created
Location: /user/42
Date: Wed, 21 Sep 2016 12:19:56 GMT
Server: Apache-Coyote/1.1
Connection: keep-alive
```

2.2.4 Modifying user entities

User entities may be modified by making a PUT request to the API URL (see introduction), appended with the ‘/user’ subroute, appended further by the User entity’s ID. For example:

```
PUT /user/42 HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
Content-Type: application/json

{
    "name": "new_username",
    "password": "some_new_strong_password",
    "fullname": "new full name",
    "email": "newemail@example.com",
    "phone": "0123456789",
    "admin": true
}
```

A successful response will have no body, but may be determined to be successful by the http status code.

```
HTTP/1.1 204 No Content
Date: Wed, 21 Sep 2016 12:19:56 GMT
Server: Apache-Coyote/1.1
Connection: keep-alive
```

Any of the listed fields will be considered as NULL, with the exception of the password field which will remain unchanged if omitted or passed NULL.

New passwords will be salted and hashed on the server side before being stored in the database.

2.2.5 Deleting user entities

User entities can be deleted by making a DELETE request to the API URL (see introduction), appended with the '/user' subroute. For example:

```
DELETE /user/42 HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

A successful response will have no body, but may be determined to be successful by the HTTP status code.

```
HTTP/1.1 204 No Content
Date: Wed, 21 Sep 2016 12:19:56 GMT
Server: Apache-Coyote/1.1
Connection: keep-alive
```

Making a DELETE request on user entity will set its inactive flag to true.
User entities which have no dependent entities may also be deleted permanently.

User entities which do have dependent entities, e.g. traps or catch entities created by the user, may be retained to maintain database integrity.

However (especially if fields storing user contact details are added at a later time), fields containing sensitive information may be voided to maintain the privacy of removed users.

2.3 Region Entities

2.3.0 Summary

Region entities represent a region in which traplines exist. They contain information about the region’s name, and the id number with which it is associated to other entities in the system.

2.3.1 Retrieving region entities

Region objects can be retrieved from the API URL (see introduction) appended with ‘/region. For example:

```
GET /region HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

An array of retrieved region objects will be returned on success, following the basic form:

```
[{
   "id": 1,
   "name": "Manawatu",
   "inactive": false,
   "can_edit": true
}]
```

By adding an id subroute to the request URL, a single region entity may be retrieved, i.e.
`GET /region/42`

By adding query parameters to a request URL, all entities (that the user has permission to access) which match the parameters may be retrieved:
`GET /region?id=42`

Available filters include:
- id
- trapline-id
- name
- inactive (only available to admins)

Filters that may be added later include:
- trap-id
- catch-id

2.3.2 Creating region entities

New region entities can be created by making a POST request to the API URL (see introduction), appended with the '/region' subroute. For example:

```
POST /region HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce

{
   "name": "Kapiti Island"
}
```

A successful response will have no body, but will contain a location header specifying an identifier with which the created resource can be retrieved:

```
HTTP/1.1 201 Created
Location: /region/42
Date: Wed, 21 Sep 2016 12:19:56 GMT
Server: Apache-Coyote/1.1
Connection: keep-alive
```

Fields storing additional details may be added at a later time, such as created_timestamp, created_user etc.

2.3.3 Modifying region entities

Region entities may be modified by making a PUT request to the API URL (see introduction), appended with the '/region' subroute, appended further by the region entity's ID. For example:

```
PUT /region/42 HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce

{
   "name": "New name of Region"
}
```

2.3.4 Deleting region entities

Region entities can be deleted by making a DELETE request to the API URL (see introduction), appended with the '/region' subroute appended further by the region entity's ID. For example:

```
DELETE /region/42 HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

Making a DELETE request on a region entity will set its inactive flag to true.
Region entities which have no dependent entities may also be deleted permanently.

Region entities which do have dependent entities, e.g. traps or catch entities existing within the region, will be retained to maintain database integrity.

It is an unlikely event that a region would be required to be deleted. Please consider offering users the option to modify it, or confirm their choice before sending a DELETE request.

2.4 Trapline Entities

2.4.0 Summary

Trapline entities represent an ordered subset of traps from the traps table, which are considered to be an independent group. The traps on a trapline will generally be checked and reset in either forward or reverse chronological order.

They contain additional identifying information as to the region, and short names that might identify the start and end of the trapline to users. Trapline entities also return the 3 most common catch types for the line for purposes of displaying in the mobile app, and default trap and bait types are also stored for pre-filling these on new traps.

2.4.1 Retrieving trapline entities

Trapline objects can be retrieved from the API URL (see introduction) appended with '/trapline.
For example:

```
GET /trapline HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

An array of retrieved trapline objects will be returned on success, following the basic form:

```
[{
    "id": 1,
    "name": "Manawatu Gorge Tawa Loop",
    "region_id": 1,
    "start_tag": "Ashhurst",
    "end_tag": "Woodville",
    "img_filename": "manawatu_gorge.png",
    "common_ct_id_1": 4,
    "common_ct_id_2": 2,
    "common_ct_id_3": 7,
    "default_bait_id": 1,
    "default_traptype_id": 3,
    "inactive": false,
    "is_admin": false,
    "can_edit": true
}]
```

By adding an id subroute to the request URL, a single trapline entity may be retrieved, i.e.
`GET /trapline/42`

By adding query parameters to a request URL, all entities (that the user has permission to access) which match the parameters may be retrieved:
`GET /trapline?region-id=13`

Available filters include:

- id
- region-id
- name
- inactive

Filters that may be added later include:

- trap-id
- catch-id

2.4.2 Creating trapline entities

New trapline entities can be created by making a POST request to the API URL (see introduction), appended with the ‘/trapline’ subroute. For example:

```
POST /trapline HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce

{
    "name": "Manawatu Gorge Tawa Loop",
    "region_id": 1,
    "start_tag": "Ashhurst",
    "end_tag": "Woodville",
    "img_filename": "manawatu_gorge.png",
    "default_bait_id": 2,
    "default_traptype_id": 3
}
```

A successful response will have no body, but will contain a location header specifying an identifier with which the created resource can be retrieved:

```
HTTP/1.1 201 Created
Location: /trapline/42
Date: Wed, 21 Sep 2016 12:19:56 GMT
Server: Apache-Coyote/1.1
Connection: keep-alive
```

Fields storing additional details may be added at a later time, such as created_timestamp, created_user, and an active flag etc.

2.4.3 Modifying trapline entities

Trapline entities may be modified by making a PUT request to the API URL (see introduction), appended with the '/trapline' subroute, appended further by the trapline entity's ID. For example:

```
PUT /trapline/42 HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce

{
    "name": "Manawatu Gorge Full Track",
    "region_id": 1,
    "start_tag": "Ashhurst",
    "end_tag": "Woodville",
    "img_filename": "manawatu_gorge2.png",
    "default_bait_id": 3,
    "default_traptype_id": 4
}
```

Omitted fields will be considered null and only the following fields may be null:

- start_tag
- end_tag
- img_filename

2.4.4 Deleting trapline entities

Trapline entities can be deleted by making a DELETE request to the API URL (see introduction), appended with the '/trapline' subroute appended further by the trapline entity's ID. For example:

```
DELETE /trapline/42 HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

Making a DELETE request on a trapline entity will set its inactive flag to true.
Trapline entities which have no dependent entities may also be deleted permanently.

Trapline entities which do have dependent entities, e.g. catch entities or damage reports existing within the traps, will be retained to maintain database integrity.

It is unlikely event that a trapline would be required to be deleted. Please consider offering users the option to modify it, or confirm their choice before sending a DELETE request.

2.5 Trapline-user Entities

2.5.0 Summary

Trapline-user entities the relationship between users and traplines. By default, unless a user is an admin, they will not have access to any traplines or the traps and data associated with them. The trapline-user entity documents an enrollment of a user to a specific trapline.

It also notes whether the user is an admin of the particular trapline.

2.5.1 Retrieving trapline-user entities

Trapline-user objects can be retrieved from the API URL (see introduction) appended with '/trapline-user.
For example:

```
GET /trapline-user HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

An array of retrieved trapline-user objects will be returned on success, following the basic form:

```
[{
    "id": 2,
    "user_id": 2,
    "trapline_id": 1,
    "admin": true,
    "can_edit": false
}]
```

By adding an id subroute to the request URL, a single trapline entity may be retrieved, i.e.
`GET /trapline-user/42`

By adding query parameters to a request URL, all entities (that the user has permission to access) which match the parameters may be retrieved:
`GET /trapline-user?trapline-id=13`

Available filters include:

- id
- user-id
- trapline-id
- admin

Filters that may be added later include:

- trap-id
- catch-id

2.5.2 Creating trapline-user entities

New trapline-user entities can be created by making a POST request to the API URL (see introduction), appended with the '/trapline-user' subroute. For example:

```
POST /trapline-user HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce

{
   "user_id": 2,
   "trapline_id": 1,
   "admin": true
}
```

A successful response will have no body, but will contain a location header specifying an identifier with which the created resource can be retrieved:

```
HTTP/1.1 201 Created
Location: /trapline-user/42
Date: Wed, 21 Sep 2016 12:19:56 GMT
Server: Apache-Coyote/1.1
Connection: keep-alive
```

Fields storing additional details may be added at a later time, such as created_timestamp, created_user, etc.

2.5.3 Modifying trapline-user entities

Trapline-user entities may be modified by making a PUT request to the API URL (see introduction), appended with the '/trapline-user' subroute, appended further by the trapline-user entity's ID. The user or trapline of an enrollment cannot be modified and a new POST or DELETE request should be made for these cases. For example:

```
PUT /trapline-user/42 HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce

{
   "admin": true
}
```

2.5.4 Deleting trapline-user entities

Trapline-user entities can be deleted by making a DELETE request to the API URL (see introduction), appended with the ‘/trapline-user’ subroute appended further by the trapline-user entity’s ID. For example:

```
DELETE /trapline-user/42 HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

Trapline-user entities have no dependent entities and will be deleted permanently.

2.6 Trap Entities

2.6.0 Summary

Trap entities represent the physical traps which get checked for catches and reset by volunteers.

They contain identifying information as to the trapline, the actual location coordinates, the type of trap and current bait being used, and when it was most recently checked etc.

2.6.1 Retrieving trap entities

Trap objects can be retrieved from the API URL (see introduction) appended with ‘/trap. For example:

```
GET /trap HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

An array of retrieved trap objects will be returned on success, following the basic form:

```
[{
    "id": 1,
    "trapline_id": 1,
    "number": 1,
    "coord_long": "-40.12345",
    "coord_lat": "175.123456",
    "traptype_id": 1,
    "status": 1,
    "created": "2016-08-16 10:35:59.123456",
    "last_reset": "2016-09-16 10:35:21.54321",
    "bait_id": 1,
    "inactive": false,
    "can_edit": true
}]
```

By adding an id subroute to the request URL, a single trap entity may be retrieved, i.e.
`GET /trap/42`

By adding query parameters to a request URL, all entities (that the user has permission to access) which match the parameters may be retrieved:
`GET /trap?trapline-id=42&region-id=13`

Available filters include:

- id
- number
- trapline-id
- trap-type-id
- status
- bait-id

Filters that may be added later include:

- catch-id
- region-id

2.6.2 Creating trap entities

New trap entities can be created by making a POST request to the API URL (see introduction), appended with the '/trap' subroute. For example:

```
POST /trap HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce

{
    "trapline_id": 1,
    "number": 1,
    "coord_long": "-40.123456",
    "coord_lat": "175.1234",
    "traptype_id": 1,
    "status": 1,
    "created": "2016-08-16 10:35:59.123456",
    "last_reset": "2016-09-16 10:35:21.54321",
    "bait_id": 1
}
```

A successful response will have no body, but will contain a location header specifying an identifier with which the created resource can be retrieved:

```
HTTP/1.1 201 Created
Location: /trap/42
Date: Wed, 21 Sep 2016 12:19:56 GMT
Server: Apache-Coyote/1.1
Connection: keep-alive
```

Fields storing additional details may be added at a later time such as created_user, etc.

2.6.3 Modifying trap entities

Trap entities may be modified by making a PUT request to the API URL (see introduction), appended with the ‘/trap’ subroute, appended further by the trap entity’s ID. For example:

```
PUT /trap/42 HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce

{
    "trapline_id": 1,
    "number": 3,
    "coord_long": "-40.6",
    "coord_lat": "175.2",
    "type_id": 2,
    "status": 0,
    "last_reset": "2016-09-16 10:35:21.54321",
    "bait_id": 1
}
```

Omitted fields will be considered null thus should be avoided.

2.6.4 Deleting trap entities

Trap entities can be deleted by making a DELETE request to the API URL (see introduction), appended with the '/trap' subroute appended further by the trap entity's ID. For example:

```
DELETE /trap/42 HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

Making a DELETE request on a trap entity will set its inactive flag to true.
Trap entities which have no dependent entities may also be deleted permanently.

Trap entities which do have dependent entities, e.g. historic catch entities or damage reports, may be retained to maintain database integrity.

2.7 Catch Entities

2.7.0 Summary

Catch entities represent a snapshot in time when its associated trap was checked and a captured animal was removed, and the trap was reset.

It contains identifying information as to the trap, the type of trap, the user who logged it, the timestamp, type of animal caught and the bait. There is also an optional field to leave a note or a picture if unusual circumstances so require it.

2.7.1 Retrieving catch entities

Catch objects can be retrieved from the API URL (see introduction) appended with '/catch'. For example:

```
GET /catch HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

An array of retrieved catch objects will be returned on success, following the basic form:

```
[{
    "id": 1,
    "trap_id": 2,
    "trap_number": 2,
    "traptype_id": 1,
    "user_id": 2,
    "catchtype_id": 1,
    "bait_id": 1,
    "note": "wow pretty big",
    "logged": "2016-09-23 08:46:58.160114",
    "img_filename": "its_massive.bmp",
    "can_edit": false
}]
```

By adding an id subroute to the request URL, a single catch entity may be retrieved, i.e.
`GET /catch/42`

By adding query parameters to a request URL, all entities (that the user has permission to access) which match the parameters may be retrieved:
`GET /catch?trapline-id=42&trap-number=1`

Available filters include:

- id
- trap-id
- trap-number
- trapline-id

Filters that may be added later include:

- region-id
- min-date
- max-date
- user-id

2.7.2 Creating catch entities

New catch entities can be created by making a POST request to the API URL (see introduction), appended with the '/catch' subroute. For example:

```
POST /catch HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce

{
    "trap_id": 2,
    "type_id": 1,
    "note": "To be fair, tis pretty big",
    "img_filename": "its_massive.bmp"
}
```

A successful response will have no body, but will contain a location header specifying an identifier with which the created resource can be retrieved:

```
HTTP/1.1 201 Created
Location: /catch/42
Date: Wed, 21 Sep 2016 12:19:56 GMT
Server: Apache-Coyote/1.1
Connection: keep-alive
```

2.8 Catch-type Entities

2.8.0 Summary

Catch-type entities represent a type of animal or state which a given trap is found in upon checking and resetting. They contain a short name, as well as an identifier for retrieving a small image associated with the caught animal type.

2.8.1 Retrieving catch-type entities

Catch-type objects can be retrieved from the API URL (see introduction) appended with '/catch-type'.
For example:

```
GET /catch-type HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

An array of retrieved catch-type objects will be returned on success, following the basic form:

```
[{
    "id": 1,
    "name": "giant rat",
    "img_filename": "a_pretty_big_rat.png",
    "inactive": false,
    "can_edit": true
}]
```

By adding an id subroute to the request URL, a single catch-type entity may be retrieved, i.e.

```
GET /catch-type/42
```

By adding query parameters to a request URL, all entities (that the user has permission to access) which match the parameters may be retrieved:

```
GET /catch-type?id=42
```

Available filters include:

- id
- name

Additional filters which may be added later include:

- catch-id

2.8.2 Creating catch-type entities

New catch-type entities can be created by making a POST request to the API URL (see introduction), appended with the '/catch-type' subroute. For example:

```
POST /catch-type HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce

{
   "name": "wild cat",
   "img_filename": "noooooooooo.png",
}
```

A successful response will have no body, but will contain a location header specifying an identifier with which the created resource can be retrieved:

```
HTTP/1.1 201 Created
Location: /catch-type/42
Date: Wed, 21 Sep 2016 12:19:56 GMT
Server: Apache-Coyote/1.1
Connection: keep-alive
```

Fields storing additional details may be added at a later time, such as created_timestamp, created_user, used_count, etc.

2.8.3 Modifying catch-type entities

Catch-type entities may be modified by making a PUT request to the API URL (see introduction), appended with the ‘/catch-type’ subroute, appended further by the catch-type entity’s ID. For example:

```
PUT /catch-type/42 HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce

{
    "name": "tiny mouse",
    "img_filename": "not_much_left.jpg"
}
```

Omitted fields will be considered null. Only img_filename may be null.

User’s should be advised against making semantic changes to catch-type entities, and instead encouraged to create new catch-type entities if there are non-trivial changes, such as a large change in size of the caught animal, or obviously a different species of stoat to the norm etc.

2.8.4 Deleting catch-type entities

Catch-type entities can be deleted by making a DELETE request to the API URL (see introduction), appended with the ‘/catch-type’ subroute appended further by the catch-type entity’s ID. For example:

```
DELETE /catch-type/42 HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

It is unlikely that a catch-type would be required to be deleted, and should only be used in the case where a catch-type entity was created in error and has not been used.

2.9 Bait Entities

2.9.0 Summary

Bait entities represent a type of bait which is used to set a trap. Bait entities are recorded by reference when a catch is recorded and a trap is reset. They contain information about the type of bait, mainly a short name and description, as well as an identifier for retrieving a small image associated with the bait type.

2.9.1 Retrieving bait entities

Bait objects can be retrieved from the API URL (see introduction) appended with ‘/bait. For example:

```
GET /bait HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

An array of retrieved bait objects will be returned on success, following the basic form:

```
[{
    "id": 1,
    "name": "chicken egg",
    "img_filename": "chicken_egg.png",
    "note": "a normal size 6 caged chicken egg",
    "inactive": false,
    "can_edit": true
}]
```

By adding an id subroute to the request URL, a single bait entity may be retrieved, i.e.
`GET /bait/42`

By adding query parameters to a request URL, all entities (that the user has permission to access) which match the parameters may be retrieved:
`GET /bait?id=42`

Available filters include:

- id
- name
- inactive (admin only)

Filters that may be added later include:

- trap-id
- catch-id

2.9.2 Creating bait entities

New bait entities can be created by making a POST request to the API URL (see introduction), appended with the '/bait' subroute. For example:

```
POST /bait HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce

{
    "name": "peanut butter sandwich",
    "img_filename": "pbj_time.png",
    "note": "a new experimental trap bait"
}
```

A successful response will have no body, but will contain a location header specifying an identifier with which the created resource can be retrieved:

```
HTTP/1.1 201 Created
Location: /bait/42
Date: Wed, 21 Sep 2016 12:19:56 GMT
Server: Apache-Coyote/1.1
Connection: keep-alive
```

Fields storing additional details may be added at a later time, such as created_timestamp, created_user, used_count etc.

2.9.3 Modifying bait entities

Bait entities may be modified by making a PUT request to the API URL (see introduction), appended with the '/bait' subroute, appended further by the bait entity's ID. For example:

```
PUT /bait/42 HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce

{
    "name": "chicken egg",
    "img_filename": "chicken_egg_2.png",
    "note": "a slightly larger size 7 caged chicken egg"
}
```

Omitted fields will be considered to be NULL. Only img_filename and note may be NULL.

User’s should be advised against making semantic changes to bait entities, and instead encouraged to create new bait entities if there are non-trivial changes, such as moving from chicken eggs to quail eggs, etc.

2.9.4 Deleting bait entities

Bait entities can be deleted by making a DELETE request to the API URL (see introduction), appended with the ‘/bait’ subroute appended further by the bait entity’s ID. For example:

```
DELETE /bait/42 HTTP/1.1
HOST: api.nestnz.org
session-token: 336b99f7-732e-4739-8af9-f1661e065dce
```

It is unlikely that a bait would be required to be deleted, and should only be used in the case where a bait entity was created in error and has not been used.