Alexander Burias

alexander.burias@gmail.com

Abstract

This document serves as a thorough documentation on Felix and Rover’s code architecture as well as the services involved in establishing secure and proper connection for the mobile front end and each administrative web application for managing records.

FELIX AND ROVER

SERVICES AND CODE ARCHITECTURE DOCUMENTATION

Table of Contents

[I. Setting up the project 1](#_Toc361845741)

[II. Setting up Development Environment. 3](#_Toc361845742)

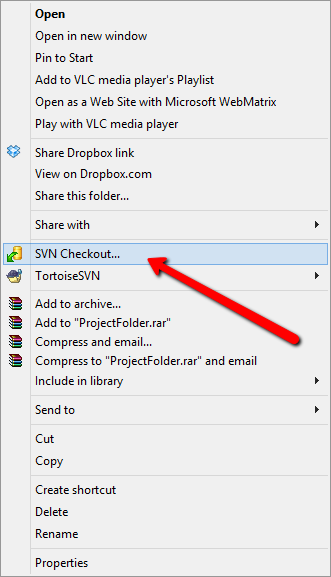
[III. Project Architecture and Relationships 5](#_Toc361845743)

[IV. Project Overviews 7](#_Toc361845744)

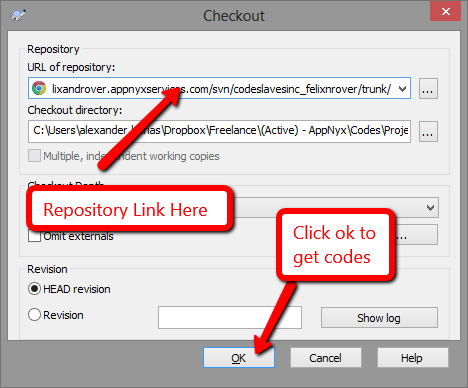
[V. Project Diagrams 19](#_Toc361845745)

# Setting up the project

1. Obtaining the code:
   1. Download and Install Tortoise SVN:
      1. <http://tortoisesvn.net/downloads.html>
      2. Select the specific download link for and install.
   2. Go to the repository and checkout the **trunk** project solution. **NOTE**: “***please ask permission from your administrator for an access to the repository****”.* If you want to just view the application structure you can check out the solution using the generic read anonymous account by just leaving everything blank in authentication or putting in ***anonymous***.
      1. Repository: <http://felixandrover.appnyxservices.com/svn/codeslavesinc_felixnrover/trunk>
      2. Create a project folder and do an SVN Checkout in that same repository.



* + 1. Click enter the correct repository link and click ok. After you get authorized you should be able to see all the codes inside the folder you just created.



* + 1. After getting the codes you should be able to see **10** project solutions.
       1. **FnR.Databases –** The main core database context.
       2. **FnR.Front –** The administrative front end.
       3. **FnR.Helpers –** Helper functions.
       4. **FnR.Mobile –** Main mobile application of felix and rover.
       5. **FnR.MobileFinal –** Redesigned UI and optimized Mobile project.
       6. **FnR.Models –** Object models mapped directly to database tables.
       7. **FnR.Repositories –** Database repository for **CRUD** functions.
       8. **FnR.VetFront –** Admin front end for each veterinary clinic.
       9. **FnR.WarehouseFront –** Admin front end for warehouse management.
       10. **FnR.WebApi –** REST based web services, optimized for mobile access.
    2. Once you have the codes in your local repository, you are now ready to setup your development environment.

# Setting up Development Environment.

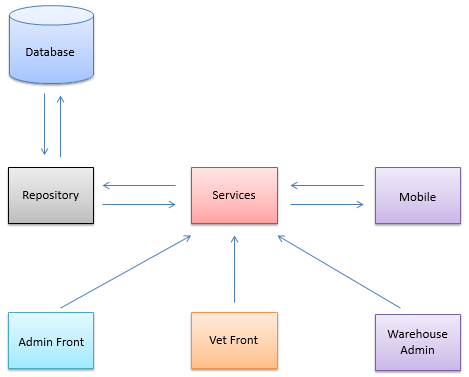
1. **Tools and Extension Downloads.**
   1. **Visual Studio 2012 –** Visual studio 2012 is the main coding environment to make the solution run. You can use Visual Studio 2012 trial version or just specific project express versions for web development and mvc 4 projects and then C# development express for class development.
      1. **Download Link -** <http://www.microsoft.com/visualstudio/eng/products/visual-studio-express-products>
      2. **Project Types:**
         1. **Web -> MVC 4 –** MVC projects are 3 Tier projects that is used in separating major concerns on development. Model, View and Controllers. These are the project types of the following felix and rover projects.
            1. **FnR.Front**
            2. **FnR.VetFront**
            3. **FnR.WarehouseFront**
         2. **Web -> MVC 4 -> Web Api –** web api projects are responsible for creating services and REST based data access and capabilities to any application that is authorized to any CRUD capabilities inside the services.
            1. **FnR.WebApi**
         3. **Windows -> Class Library –** Class library is a basic class project, this is mostly used in creating models, helper classes as well as the database context.
            1. **FnR.Databases**
            2. **FnR.Helpers**
            3. **FnR.Models**
            4. **FnR.Repositories**
         4. **Web Project –** Web project is a standalone project in developing HTML, CSS, Javascript based application as well as mobile applications.
            1. **FnR.Mobile**
            2. **FnR.MobileFinal**
   2. **MS SQL Server 2008 / 2012 –** MS SQL Server is a Relational Database application that will hold any data related concerns for the felix and rover application. This is our main database application that is connected directly to the models and database classes.
      1. **Download Link -** <http://www.microsoft.com/en-us/download/details.aspx?id=1695>
      2. **Tools**
         1. **SQL Server Management System –** This specific tool is useful when you want to manage the database directly without having to go through the code generator and database mapper.
            1. **Download Link -** <http://www.microsoft.com/en-us/download/details.aspx?id=8961>
2. **Useful Tools and Development Extensions (Optional).**
   1. **Resharper –** Rapid application development helper to enhance and speed up code development by providing key mappings, auto correct and advanced code intellisense and many more.
      1. **Download Link -** <http://www.jetbrains.com/resharper/download/>
   2. **EntityFramework Powertools –** is a visual studio 2012 extension that offers a set of tools to develop and manipulate object mapper capabilities which offers more enhanced diagramming tool and reverse engineering for any existing database as well as generating database views instantly based on your models and database classes.
      1. **Download Link -** <http://visualstudiogallery.msdn.microsoft.com/72a60b14-1581-4b9b-89f2-846072eff19d>

# Project Architecture and Relationships

Felix and Rover project is a multi-tier application that revolves around a solid database structure that is being accessed by another layer called repositories which is the only part of the project that has solid CRUD (Create, Read, Update, and Delete) access into the database models.

This is then being handled and manipulated according to needs with a final data access layer called services. Services are REST based secure web api application that will be in charge of all business logic and business rules that requires any access to the data model or the database. This central services are accessed by every application or any application that is given authorization by the service weather it’s a web application, an mvc application or just plain mobile application.

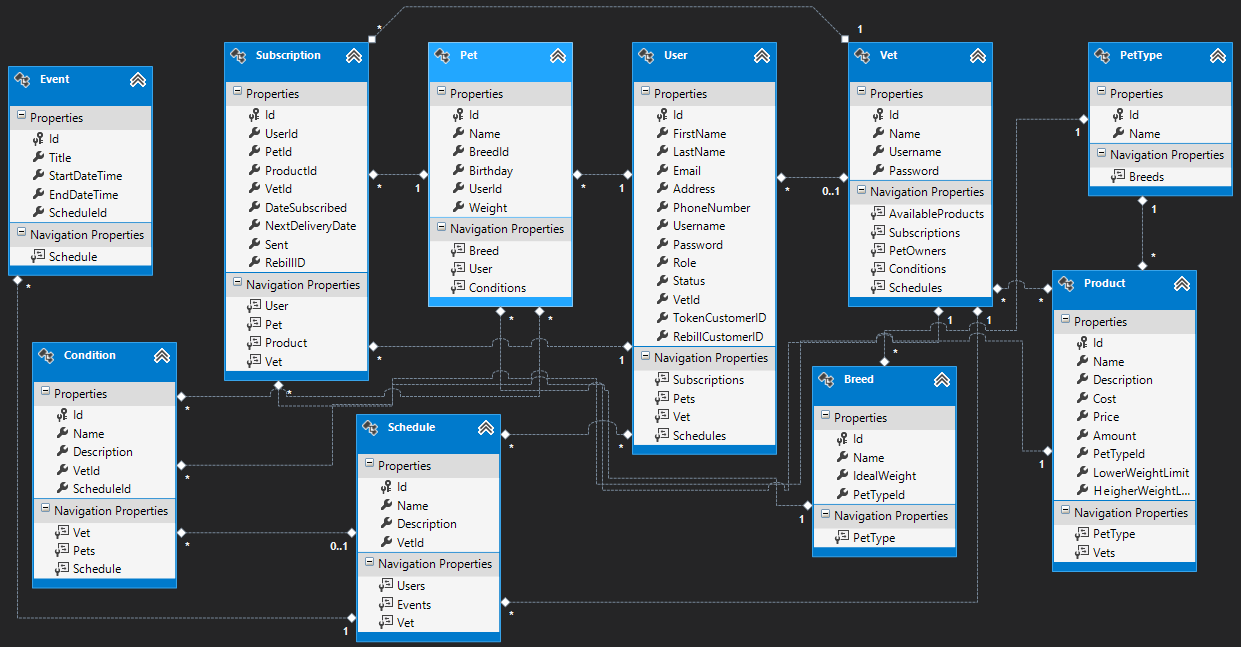
* 1. **Architecture Diagram**

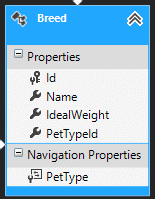
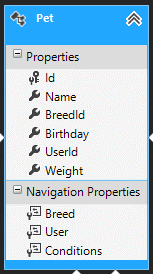
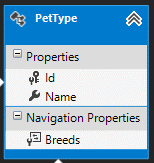
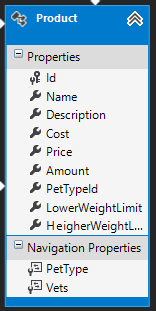
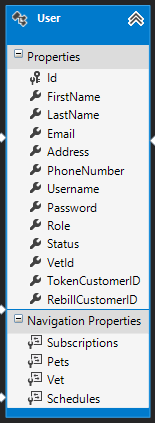
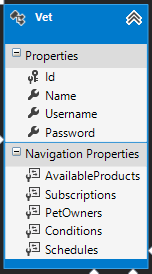
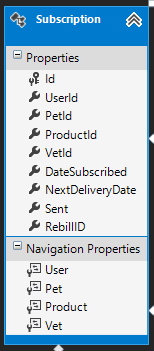
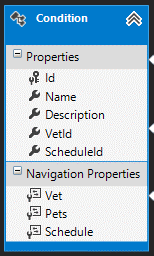
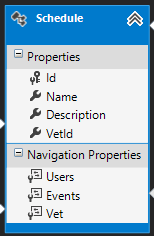
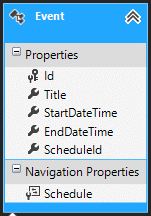
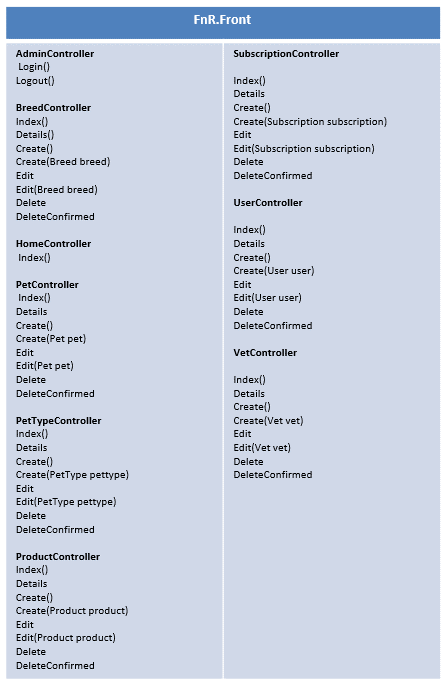


* 1. **Architecture Parts**
     1. **Database –** Centralized data store for the whole application and its models.
     2. **Repository –** Data access functions for Creating, Updating, Reading and Deleting objects in the database. The only interface between the application project and the actual data storage.
     3. **Services –** Business logic interface of the whole application, In charge of almost any tasks that needs business related access to the database, like filtering, manual storage and other business logic specific to each and every application.
     4. **Mobile –** Mobile application, built with only HTML, CSS and Javascript, it is the main interface that users/clients or pet owners see to access their account, update it and subscribe to any product that are available to any vet.
     5. **Admin Front –** Administrative front end, an MVC 4 application that will cater almost anything not only CRUD capabilities for the database but also business related logic needed for anything to function.
     6. **Vet Front –** Veterinary front end is an administration for vets. This is where they handle almost everything related to their clinic, setting schedule, setting clients, users as well as advanced features like subscriptions and un-subscription manually.
     7. **Warehouse Admin –** A warehouse (item storage / local storage) is the front end administration for handling products and product related queries. This is also where they check their generated income and set items as being sent or not, basically a front end to handle product deliveries as well as product backlogs and everything product related.

# Project Overviews

* 1. **FnR.Databases –** A database context project in charge of mapping database objects into the model classes. This is basically the schema of the whole project.



* + 1. **Attributes and Properties –** All the project attributes are mapped to a specific table inside the database.
       1. **Breeds –** Mapped to Breeds table.
          1. 
       2. **Pets –** Mapped to Pets table in our schema.
          1. 
       3. **PetTypes –** Mapped to PetTypes table.
          1. 
       4. **Products –** Mapped to Products table.
          1. 
       5. **Users –** Mapped to Users table.
          1. 
       6. **Vets –** Mapped to Vets table.
          1. 
       7. **Subscriptions -** Mapped to Subscriptions table.
          1. 
       8. **Conditions –** Mapped to Conditions table.
          1. 
       9. **Schedules –** Mapped to Schedules table.
          1. 
       10. **Events –** Mapped to Events table.
           1. 
    2. **Project Class Diagram**
       1. 
  1. **FnR.Front –** This is basically everything packaged into one project, an administration application for managing all records in the database as well as managing business logic. It is an MVC Project which stands for Model View Controller project, a separation of concern for almost all multi-tier application.
     1. **Controllers**
        1. **Admin Controller –** Administrative controller for mostly user authorization.
           1. **Login –** Login specific user.
           2. **Logout –** Logout currently logged in user.
        2. **Breed Controller –** Breed specific controller, managing breeds, creating and linking to a specific pet type.
           1. **Index**
           2. **Details**
           3. **Create**
           4. **Edit**
           5. **Delete**
        3. **Home Controller**
           1. **Index**
        4. **Pet Controller**
           1. **Index**
           2. **Details**
           3. **Create**
           4. **Edit**
           5. **Delete**
        5. **PetType Controller**
           1. **Index**
           2. **Create**
           3. **Edit**
           4. **Delete**
        6. **Product Controller**
           1. **Index**
           2. **Details**
           3. **Create**
           4. **Edit**
           5. **Delete**
        7. **Subscription Controller**
           1. **Index**
           2. **Details**
           3. **Create**
           4. **Edit**
           5. **Delete**
        8. **User Controller**
           1. **Index**
           2. **Details**
           3. **Create**
           4. **Edit**
           5. **Delete**
        9. **Vet Controller**
           1. **Index**
           2. **Details**
           3. **Create**
           4. **Edit**
           5. **Delete**
     2. **Project Class Diagram**
        1. 
  2. **FnR.Helpers –** Series of helper classes in managing string manipulations for authentication as well as sending emails for notification and anything related. Helper is just a basic C# Class library project with a bunch of custom classes for each helper type like string and email.
     1. **Classes**
        1. **Email Helper**
           1. **SendEmail**
           2. **SendEmailWithAttachment**
  3. **FnR.Mobile –** Is the main mobile application for the whole felix and rover system. It’s a plain HTML, CSS, Javascript project that links up all the functions into the Services.
  4. **FnR.Models –** Is also a C# class library project that is a 1 to 1 mapping of the database tables into these classes. For more information, please check the project **FnR.Databases** for a detailed information regarding the fields and classes that relates to all the models. All these models are being mapped through the database by a datacontext making them datasets and then directly into a table inside the schema from MS SQL Server RDBMS (Relational Database Management System).
  5. **FnR.Repositories –** Another C# class library that creates repository interfaces for each object into the table records. A main class called **Repository<T>** will get inherited having a type of **T** that will then be mapped to any of the objects or model classes. So all the functionality will get inherited by Model Specific Repositories and the CRUD functions will know where to do its updates into which tables.
     1. **Repository<T>**
        1. **GetEntityByKey –** Retreives a specific record by its **Id**
        2. **Get –** Retrieves all record that is of type **T.**
        3. **CreateEntity –** Create a new record in the database of type **T.**
        4. **UpdateEntity –** Updates a record that has been found in the database.
        5. **Delete –** Delete an entry from the database found by the provided **Id.**
     2. **Project Class Diagram**
  6. **FnR.VetFront –** Administrative front end for all veterinary clinics. In charge of managing all information regarding a specific vet that is currently logged in. This is where they manage what products they provide to which pet and which users they want and they can also create subscriptions to any specific users into that vet clinic. This is an MVC 4 Project.
     1. **Controllers**
        1. **HomeController**
           1. **Index**
           2. **Login**
           3. **Logout**
        2. **IncomeController**
           1. **Index**
        3. **PostVisitCareController**
           1. **Global Variables**

**\_vetRepo**

**\_conditionRepo**

**\_userRepo**

* + - * 1. **Controller Functions**

**Index**

**AddCondition**

**EditCondition**

**DeleteCondition**

**GetPetCondition**

**AddPetCondition**

**RemovePetCondition**

* + - 1. **ProductController**
         1. **Global Variables**

**\_repo**

**\_productRepo**

* + - * 1. **Controller Functions**

**Index**

**Create**

**RemoveProduct**

**GetProduct**

* + - 1. **SchedulingController**
         1. **Index**
         2. **Create**
         3. **Edit**
         4. **Delete**
         5. **CreateEvent**
         6. **CreateSchedEvent**
         7. **DeleteEvent**
         8. **Unassign**
         9. **AssignSchedule**
      2. **SubscriptionController**
         1. **Global Variables**

**\_repo**

**\_subRepo**

**\_userRepo**

**\_productRepo**

* + - * 1. **Controller Functions**

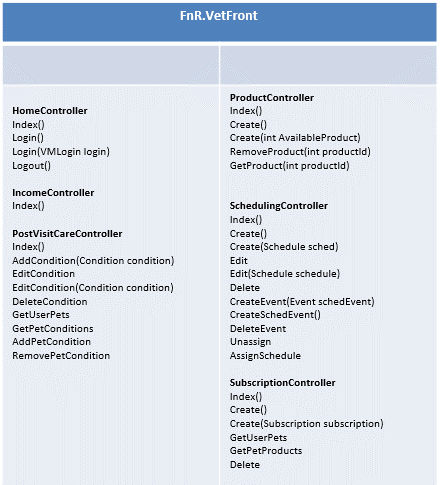
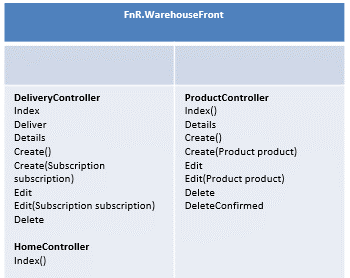
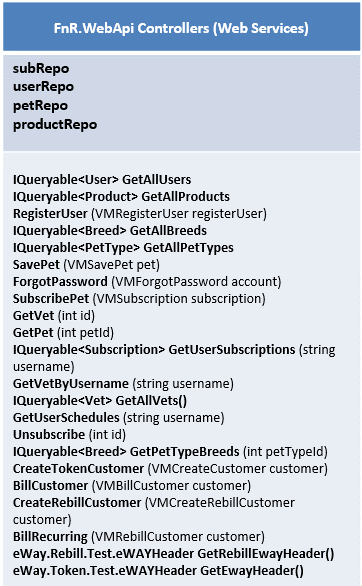
**Index**

**Create**

**GetUserPets**

**GetPetProducts**

**Delete**

* + 1. **Project Class Diagram**
       1. 
  1. **FnR.WarehouseFront –** An mvc 4 project in managing all warehouse related projects, product as well as subscription and delivery management.
     1. **Controllers**
        1. **Delivery Controller**
           1. **Index**
           2. **Deliver**
           3. **Details**
           4. **Create**
           5. **Edit**
           6. **Delete**
        2. **HomeController**
           1. **Index**
        3. **ProductController**
           1. **Index**
           2. **Details**
           3. **Create**
           4. **Edit**
           5. **Delete**
     2. **Project Class Diagram**
        1. 
  2. **FnR.WebApi –** This is a web api project that provides the core business logic interfaces for all connected and authorized application. This is a REST based web services that takes in any object and returns json or any specific format based on database models or objects.
     1. **FelixAndRoverController**
        1. **Global Variables and Repositories**
           1. **\_subRepo**
           2. **\_userRepo**
           3. **\_petRepo**
           4. **\_productRepo**
        2. **Available Services**
           1. **GetAllUsers**
           2. **GetAllProducts**
           3. **RegisterUser**
           4. **GetAllBreeds**
           5. **GetAllPetTypes**
           6. **SavePet**
           7. **ForgotPassword**
           8. **SubsribePet**
           9. **GetVet**
           10. **GetPet**
           11. **GetUserSubscriptions**
           12. **GetVetByUsername**
           13. **GetAllVets**
           14. **GetUserSchedules**
           15. **GetPetTypeBreeds**
           16. **CreateTokenCustomer**
           17. **BillCustomer**
           18. **CreateRebillCustomer**
           19. **BillRecurring**
           20. **GetEwayRebillHeader**
           21. **GetEwayHeader**
     2. **Graph Diagram**
        1. 

# Project Diagrams

