Group Project

Project Darknet – Progress Report

Johann Beleites, Ibtehaj Nadeem, Farah Kiran Patel,
Josh Treon, Augustin Zidek

University of Cambridge, Lent 2014, Version 2014-02-13 01:14:37

Contents

1. Design Changes

Since writing the specification, we have made a few design changes in order to expand the extensibility of our system. The first change is that instead of a `DataCollector` that will both collect data from a source and provide it to the `DataHandler`, we have separated these tasks into two separate classes: `DataCollector`s which collect data from a primary or secondary source to be put in the database and `DataProvider`s which retrieve the data relevant to a specific source and pass it to the effect that requires the data.

In order to maintain separation between these classes and the database we have also created a `DatabaseManager` which handles all of the direct interactions to the database so when a `DataCollector` has information to store it passes the data to the `DatabaseManager` which then stores it correctly in the database. We also added a `DatabaseProvisionManager` which will serve as the layer of separation between the `DatabaseManager` and `DataProviders`. We also added a `DataCollectionManager` class that will start the correct data collectors in the correct order so there is only one class that needs to be started manually by the user.

The new class diagrams included in the Appendix reflect these changes. There are two changes that are not reflected in the diagrams. The first is that we now are no longer using the Spektrix API because it is easier for them to just send us a CSV file with the necessary primary data. Additionally, since image files cannot be stored in just the database, we added image storage. The second recent change is in how we store and retrieve data in the database. Instead of having separate tables for each `DataCollector` we now have separate tables for each attribute. This allows us to simplify data provision because now instead of a `DataProvider` for each source we only require one that can access all the information.

2. Progress

We are currently implementing phase 1 of our Project Plan (reproduced in the Appendix of this progress report).

The distribution of work for this first phase is as follows:

- **Augustin:** Backend GUI, Image Storage, documentation
- **Farah:** `PrimaryDataCollector`, `ManualInputDataCollector`
- **Ibtehaj:** database, `DatabaseManager`, `DataCollectionManager`, interfaces, `Individual`, `Properties`
- **Johann:** research on Man-In-The-Middle Attack, `FacebookDataCollector`
- **Josh:** `SecondaryDataCollector`, `FacebookDataCollector`

So far we have completed implementation of the database and image storage as well as the interfaces, and `Individual` and `Properties` classes. Each of these has been tested independently by its creator.

The `ManualInputDataCollector` and `PrimaryDataCollectorGUI` (previously referred to as Backend GUI) are near completion and will be implemented and tested by the second progress meeting.

Progress on `SecondaryDataColletors` has proven to be more difficult than initially anticipated. In doing research on the Man-In-The-Middle Attack we are questioning the legality of some of the aspects of this method of data collection. Additionally, when we went to start implementing the `FacebookDataCollector` we came across some terms and conditions of the Facebook API that indicate searching for Facebook users by name or email address (the only search criteria we have access to) is in fact violating the terms and would therefore might not be legal.

In fact we found this quote from a Facebook representative "This is by design, searching for users is intended to be a user to user function only, for use in finding new friends or searching by email to find existing contacts on Facebook. The 'scraping' mentioned on StackOverflow is specifically against our Terms of Service `https://www.facebook.com/terms.php` and in fact the only legitimate way to search for users on Facebook is when you are a user." We have considered creating a fake user for this purpose, but at it seems to be muddy legally we have chosen to defer so that we can discuss this issue at our client meeting and if need be raise it with the project organisers.

In order to have some secondary data available for the meeting we have started on the `TwitterDataCollector` but at the Twitter API is less used it might take longer to implement. We also need to look into the legality of searching using name and email using its API. Additionally the authentication process required to use the API is also posing some problems. The goal is to have a version of this working by Friday, but it may not be feasible.

If the legality of using social networks is a great concern, we will need to consult with our client and perhaps the project organisers in order to decide what alternate sources we can use legally for the purposes of this project.

3. Appendix

3.1 Project Plan

3.1.1 Phase 1 – Implement the basics of the backend.

This includes the following modules: the database, `DataCollector` interface, `PrimaryDataCollector` abstract class, `SecondaryDataCollector` abstract class, `ManualInputDataCollector`, `SpektrixDataCollector`, `FacebookDataCollector`, `DatabaseManager`, `DataCollectionManager`, `Individual`, `Properties`, Backend GUI, Image storage

Goal of phase: to have database up and running and at least one primary and one secondary DataCollector implemented as well as the modules that connects them so that we can show the client what types of data are feasible to retrieve

Time Estimate: 1.5 weeks

3.1.2 Phase 2 – Implement more backend and start implementation of frontend and connect the two.

This includes the following modules: `TwitterDataCollector`, `DataProvider` interface, `PrimaryDataProvider` class, `SecondaryDataProvider` abstract class, `FacebookDataProvider`, `DataProvisionManager`, `EffectExecutionGUI`, Effect abstract class, `ReportEffect`

Goal of phase: Have a working prototype system that can retrieve Manual Input, run a single `DataCollector`, store and access the data correctly and use the `DataProvisionManager` to provide input for the `ReportEffect` which should generate basic report of all data found

Time Estimate: 2 weeks

3.1.3 Phase 3 – Add additional DataCollectors and Effects

This includes: `TwitterDataProvider`, `ManInTheMiddleDataCollector`, `ManInTheMiddleDataProvider`, `PictureWallEffect` and others

Goal of phase: To expand our prototype system to include more sources of data and more effects. NOTE: this phase will only be undertaken in the case we have sufficient time before the deadline to reliably complete it

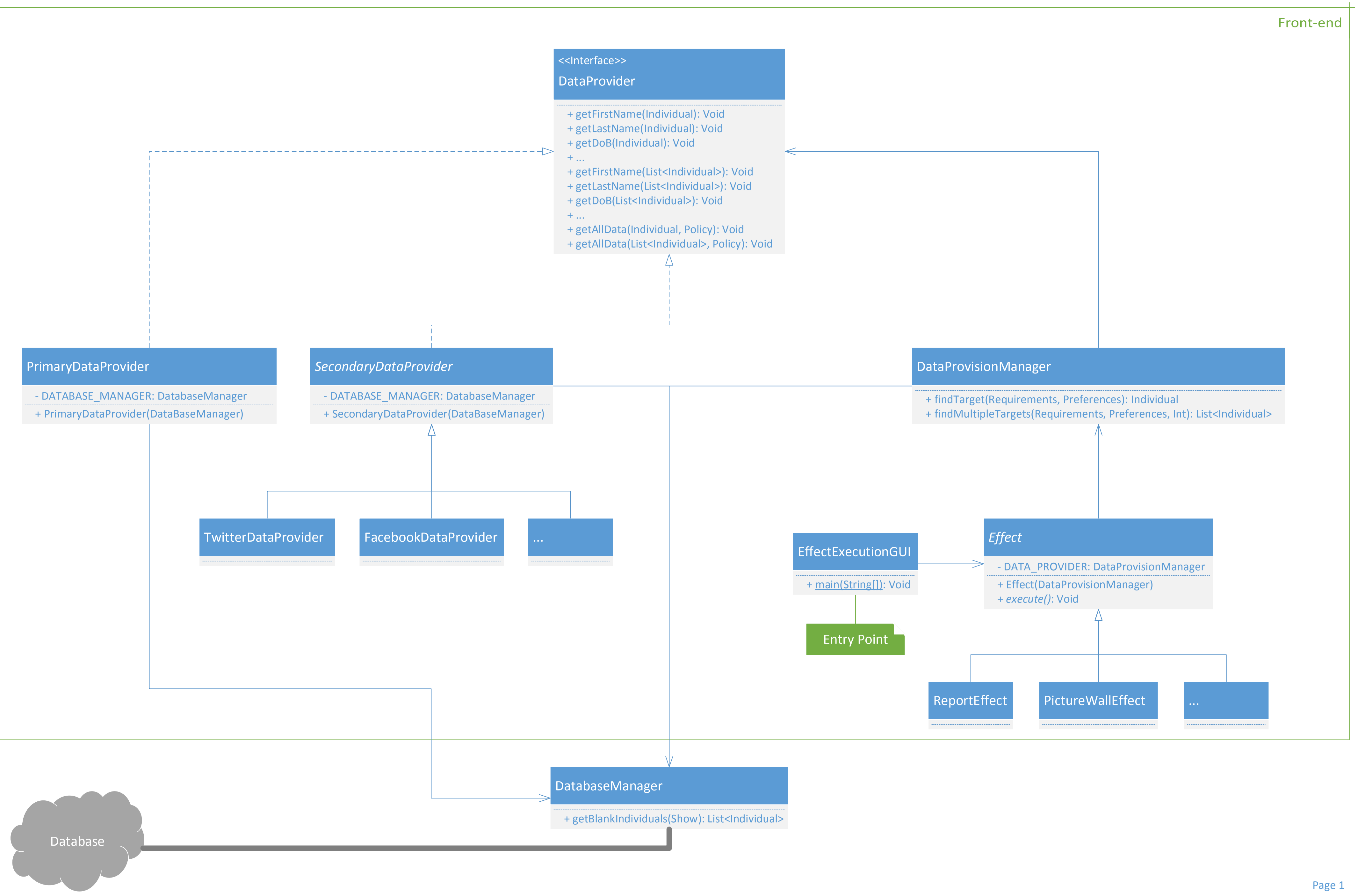
Time Estimate: 1 week

NOTE – allocation of modules to implement will be decided at the beginning of each phase. The distribution of modules will work on a preferential basis and be as evenly spread out as possible.

3.2 Class Diagrams

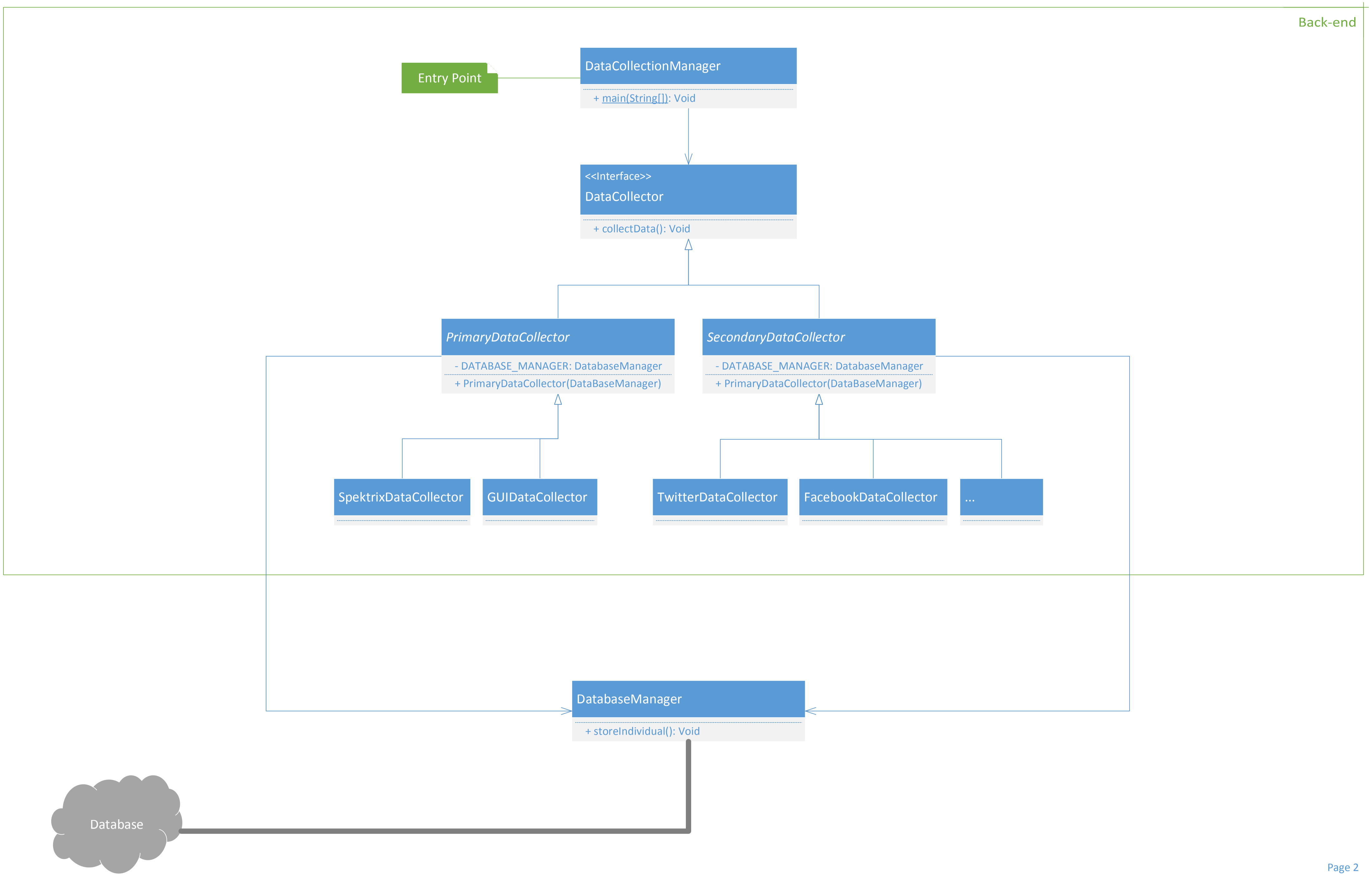
Project Darknet — Class Diagram

February 1, 2014


Front-end
<<Interface>>
DataProvider
+ getFirstName(Individual): Void
+ getLastName(Individual): Void
+ getDoB(Individual): Void
+ ...
+ getFirstName(List<Individual>): Void
+ getLastName(List<Individual>): Void
+ getDoB(List<Individual>): Void
+ ...
+ getAllData(Individual, Policy): Void
+ getAllData(List<Individual>, Policy): Void
PrimaryDataProvider
- DATABASE_MANAGER: DatabaseManager
+ PrimaryDataProvider(DataBaseManager)
SecondaryDataProvider
- DATABASE_MANAGER: DatabaseManager
+ SecondaryDataProvider(DataBaseManager)
DataProvisionManager
+ findTarget(Requirements, Preferences): Individual
+ findMultipleTargets(Requirements, Preferences, Int): List<Individual>
TwitterDataProvider
FacebookDataProvider
...
EffectExecutionGUI
+ main(String[]): Void
Entry Point
Effect
- DATA_PROVIDER: DataProvisionManager
+ Effect(DataProvisionManager)
+ execute(): Void
ReportEffect
PictureWallEffect
...
DatabaseManager
+ getBlankIndividuals(Show): List<Individual>
Database

Project Darknet — Class Diagram

February 1, 2014


Back-end
Entry Point
DataCollectionManager
+ main(String[]): Void
<<Interface>>
DataCollector
+ collectData(): Void
PrimaryDataCollector
- DATABASE_MANAGER: DatabaseManager
+ PrimaryDataCollector(DataBaseManager)
SecondaryDataCollector
- DATABASE_MANAGER: DatabaseManager
+ PrimaryDataCollector(DataBaseManager)
SpektrixDataCollector
GUIDataCollector
TwitterDataCollector
FacebookDataCollector
...
DatabaseManager
+ storeIndividual(): Void
Database