# Investigator

### investigate your players' tracks

by Lukas Höppner and Gregor Čepin for Design of Dynamic Web Systems at the Technical University of Luleå.

- http://investigator.meteor.com/
- https://github.com/DynamicGameAnalytics/Investigator

## Introduction

*Investigator* is a tool to collect and visualize player data for games. It can be used to get realtime information about player activity for multiple games and analyze this data later on.

## Features

- Creation of different games
- Multiple sessions for each game
- Simple event triggering via RESTful HTTP api
- Easy installation thanks to meteor
- Admin interface with access to raw data
- MongoDB storage
- HTML5 frontend with realtime updates
- Login via Password, Google+, Facebook, Twitter or GitHub

## Installation instructions

Inspector is based on meteor, which can be found at https://www.meteor.com/. Meteor requires OS X or Linux. After installing meteor and checking out the repository, just run `meteor run` from the repository root directory. This will start the application at http://localhost:3000.

## Technology

Investigator uses meteor with some extra packages.

- **CollectionFS**: binary file uploads and storage
- **Meteor-Bower**: automatically installing bower packages
- **Houston**: drop-in admin interface
- **Autoform**: quick form generation
- **Iron-Router**: application routing with browser history handling
- **Meteor-Collection2**: collection schemes and validation

The frontend is based on **web components** and the **Polymer project**.

## Application design

### Models

The application is based on three main models: Games, Sessions and Events.

Games are the first layer of separation between different groups of data. Games can be shared between users to give other developers access to the data for reading and writing. Every game has a unique token which is used to create sessions for this game.

Sessions are the second layer of separation between different groups of data. Sessions are used to assign events to a specific user (or a

*session* of a user) to be able to track sequences of events for this user. With this sequence of events it is possible to analyze in which order the events occured. Every session has a unique session token which is used to trigger events for this session.

Events are the main data instances. Every belongs to a session, has a type, a creation timestamp and may have some extra data. The event types can be different from game to game, so every game can define its own set of possible events by simply sending different type-strings when an event occurs. Events can be sent via a simple HTTP-api from every internet-enabled game or device.

# Security aspects

The collected data is only readable for the owner to keep the privacy, but to be able to trigger events from every device via the HTTP api, this api has to be open for everyone. It would be possible to generate a secret for each game that is required to trigger events, but due to reverse engineering and network traffic analysis, it is impossible to keep that secret secret. It is still necessary to ship the game token with the game and if an attacker gets hands on this token, he can still access every api method, but he has to generate a new session and this session keeps the attacker in a relatively small area of possible damage. If an corrupted session is found, it can simply be deleted or ignored from the analysis. However, the value of an attack on this kind of systems is very low to non-existing.

# Known Problems

Two major problems occured during the development. The first one is meteor itself. Meteor itself is really great and does a lot of things for you, but it also does hide a lot of things. You can use bower for example, but if you do so it is not possible to host the application on the meteor server, because it only runs `meteor run` and this doesn't install the bower packages. So you need to install a package for meteor that installs the bower packages, but if the newest version of the package in the meteor package repository is to old, it is really hard to install a newer version of it via git. We solved this by adding a submodule of a fixed fork of the package to our repository and use this version instead. All in all there is an additional layer between the original package (like bower) and your application and this layer has to be updated which causes another delay in the development circle.

The second major problem are cross-origin requests to the api from a browser and PUT data in these requests. When using a third-party api client like *Postman* for the requests everything works fine, but as soon as this requests are sent by a browser, the message body doesn't get parsed. And, again, you don't know why, because meteor and its packages handle the raw request. As a temporary workaround, the application offers an additional GET method for triggering events, but this method lacks the possibility of providing additional event data.

# Future work

Currently there is only one realtime graph to display all events, but it is possible to add additional graphs like a sankey diagram to visualize player *movements* or other graphs for further data analysis.

In addition to tracking game events, a model for tracking game errors is prepared and may be implemented in the future to make it possible to track game crashes and scripting errors and maybe automatically creating bug reports.

Furthermore there is much space for analysis improvements, such as different event filters and reports simmilar to Google Analytics.

# API documentation

### Get session token

`GET /session_token/:game_token` : Generates a new session token for the given game and replies with

```
{
  "session_token": session_token
}
```

### Trigger event

`GET /event/:session_token/:event_type` : Triggers a event with the given type for the given session and replies with

```
{
  "game": game_token,
  "session": session_token,
  "event": {
    "type": event_type
  }
}
```