Anti-Cheat: monitoring

Anti-Cheat system main goal is to counter hacking and cheating. The working principles of anti-cheat are very similar to antiviruses, they scans memory searching for signatures of malicious code, observes if files haven't been tampered with. Those tasks are tightly coupled with operating system and can not be implemented in JVM. However monitoring of system, like getting list of running processes or taking screenshots, is achievable. The task is to implement basic infrastructure of Anti-Cheat system with monitoring.
Server should track connected users and fetch information from client. Also it should perform authentication and authorization.
Client application should gather needed information: the list of processes, screenshot. User should be able to create an account and login with it.
Administration panel to view information of users, warn or ban them.

Infrastructure

Server can run on any machine which has JavaEE Application Server. It has been developed and tested using Payara Server 4.1.1.164 and MariaDB 10.1.
Client requires JVM and Windows OS due necessity of native access to operating system.
Administration panel consists of API and Web application. API requirements are the same as the server's. Web application requires any modern browser or IE9+.
Performance testing was carried out using Dell laptop with i5-4200U CPU @ 2.3Ghz, 8GB RAM. Test were run using Node.Js.

Programming technologies

Client

Java SE - main application
Java FX - graphical interface
Java Native Access (JNA) - gathering information about system
RMI - communicating with server

Server

JavaEE - based on
EJB - business logic
JPA(eclipselink) + MariaDB Connector/J - database libs
RMI - communicating with client
Web Applications - serving of web assets
Jersey(JAX-RS) - RESTful Web services
Tyrus - WebSockets
JJWT - JSON Web Token for Java

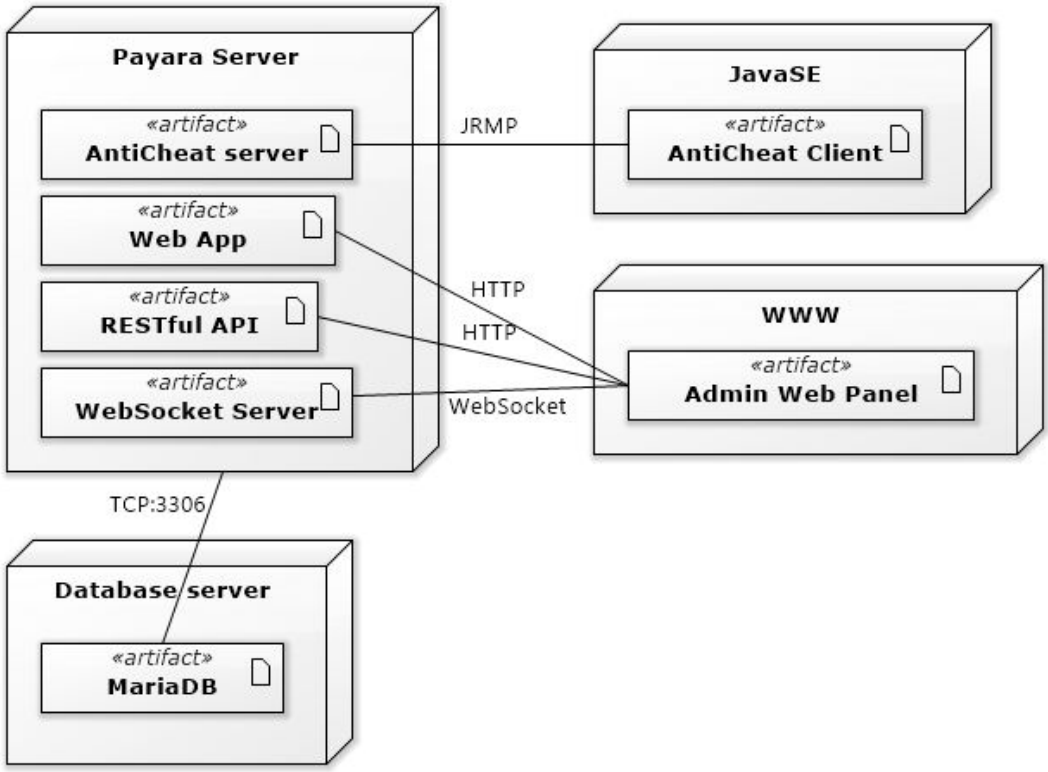
Jackson - JSON Processor

WEB Admin panel

ES6 - standard of JavaScript
Angular2 - web development platform
Ng2-restangular - REST API client
SCSS - CSS extension language
Material2 - design

Architecture

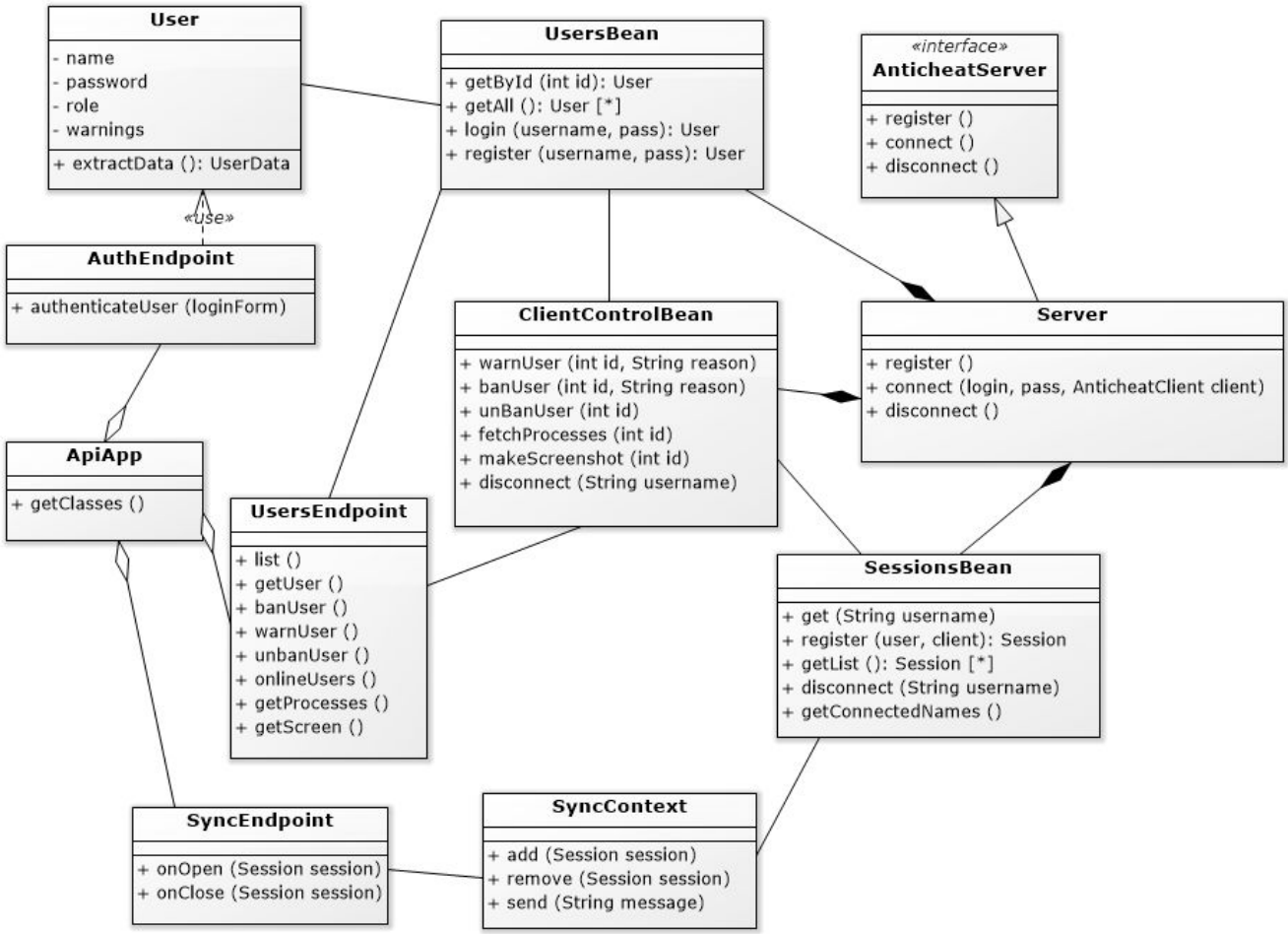
System consists of four parts: anti-cheat backend, client, admin web panel and database.



AntiCheat backend

AntiCheat backend is served by JavaEE Application server - Payara server. Backend serves AntiCheat server, web application, RESTful API and WebSocket Server.
AntiCheat server is responsible for communication with AntiCheat client. It provides registration of a new user, authorization, tracking of sessions.

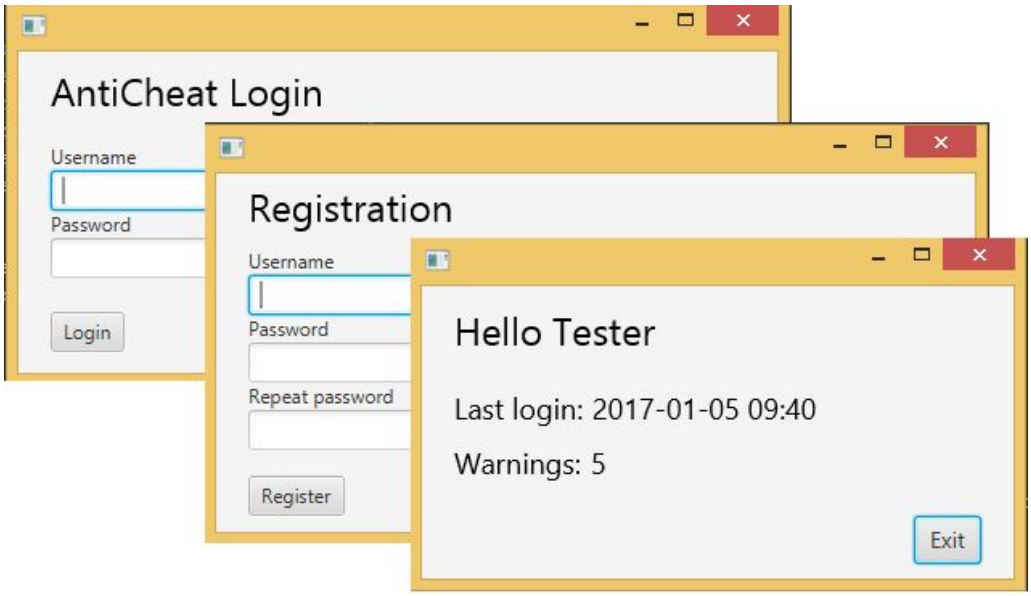
RESTful API provides user information, list of connected users and control of available client actions (get screenshot, process list) and management actions(warn, ban). API has authentification and authorisation of admin users.
Web app serves static web assets - html, javascript, css, etc.
WebSocket server is used to provide events, for example inform that user connected or disconnected.
Server communicates with client via AnticheatClient remote object, which is passed as parameter during login to server by calling connect to Server remote object. User information is managed by UsersBean. Connected clients are tracked by SessionBean. API endpoints are defined in AuthEndpoint and UsersEndpoint. API uses ClientControlBean to perform actions related to connected client. Websocket endpoint is defined in SyncEndpoint.



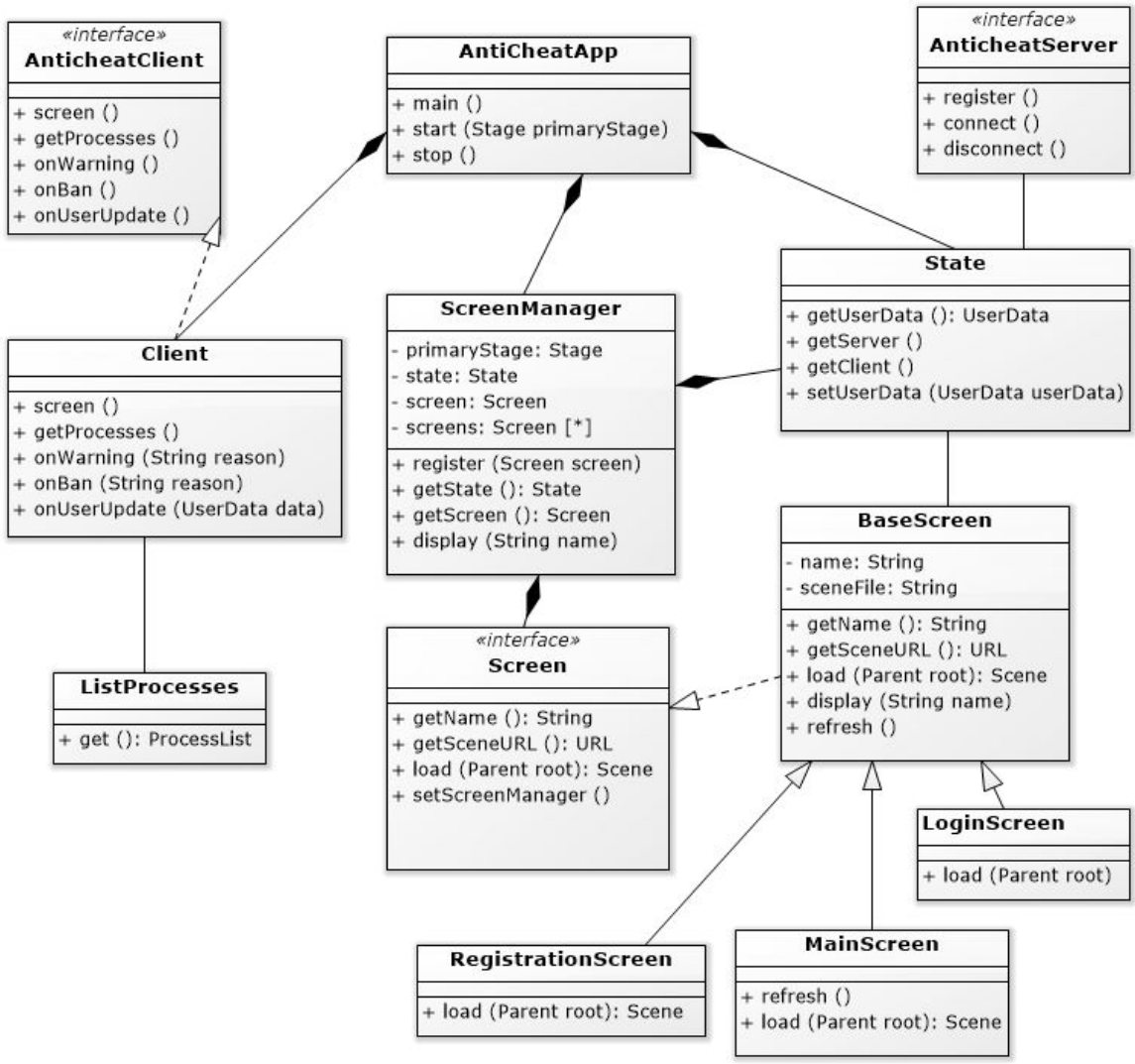
AntiCheat client

Client is JavaSE application and it communicates with server using RMI. Client can provide data about user's system upon request for example list of running processes or screenshot. Client works only on Windows, due necessarily to access internal system objects using JNA. Application provides registration, login and display of user details functionalities.

Here is screens of application:



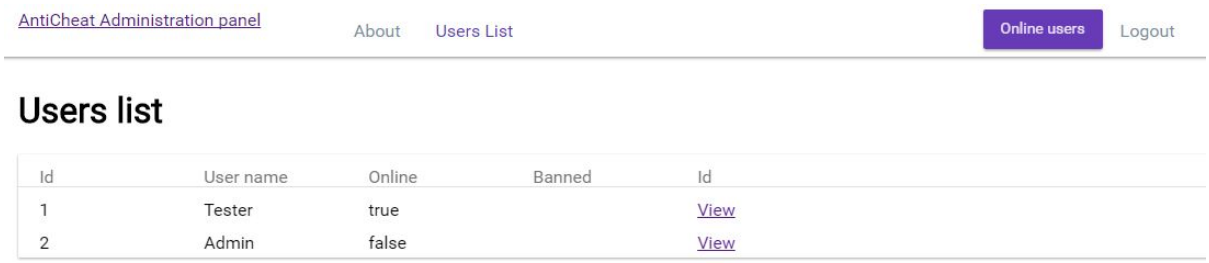
Client GUI is controlled by ScreenManager class, which loads and activates screens. Screen class is responsible for logic of one window, for example registration GUI logic is defined in RegistrationScreen class. Data to screen is passed via State class, this state is shared across whole application and chan be updated by server.
Client communicates with server using AntiCheatServer remote object, which is discovered using Naming Service. In this system server is active part which initiates actions on client using Client class. Also server invokes callbacks to notify about warning, ban or changes of data.



Admin Web Panel

Administration Panel is single page application build on Angular2 web development platform. Admin Web Panel communicates with server using RESTful API. API endpoints are protected, JWT is used for authentication and authorization. Data is transferred in JSON format. WebSockets are used for listening to events, for example track connected and disconnected users.

Examples of admin interface:



User Tester

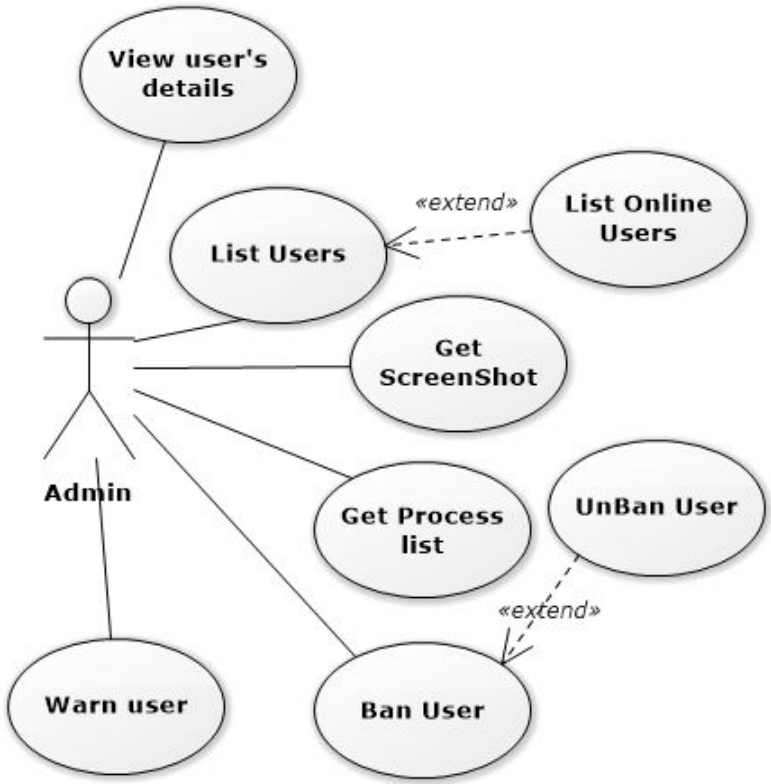
Role
    user
Last login
    1/10/2017, 7:15 PM
Warnings
    5
Online
    Yes
Status
    Active

Actions

Get screenshot | Get process list

Warn | Ban

Admin panel lets to list all available/online users, view user's details, ban/unban/warn user, get list of running processes in user's system and make a screenshot of user's screen.



Installation

AntiCheat application requires no specific installation, just run as regular Java SE application.

AntiCheat Server requires database configuration and application deployment to JavaEE application server. Instructions how to do that are available online: http://blog.payara.fish/how-to-deploy-an-application-on-payara-server-or-glassfish

Admin Web Panel files are already prepared for deployment, however here are full instructions:

1. Download the latest NodeJS: https://nodejs.org/dist/v6.7.0/node-v6.7.0-x64.msi
2. Install Gulp - tasks runner: `npm install -g gulp`
3. Install dependencies: `npm install`
4. Prepare deployment files: `gulp build`
5. Copy files to web server root.