CSC7325 Project

Notification System (Push + Messaging)

CSC7325 Project

Notification System (Push + Messaging)

A communication system between students and professors, implemented in Java Web Services and mobile frameworks

Goals

Goals

Goals

Web Application (Tomcat):

- See registered student + Device Info.
- Delete students and/or devices.
- Send messages (Normal + Push).
- See sent messages

Goals

Web Application (Tomcat):

- See registered student + Device Info.
- Delete students and/or devices.
- Send messages (Normal + Push).
- See sent messages

Mobile Application

- Register to the service with e-mail confirmation
- See received messages
- Receive push notifications
- Local database

Server-side technologies

- Server - Independent Architecture (But tested on Tomcat)
- Java 1.6 + JDBC + Mysql
- MySQL Connection data pool (BoneCP)
- Barebones JSP 2.0 (with JSTL + templating support)

- JSON for the REST API
- Eclipse + MAVEN
- Git for source management

Client-side technologies

- iOS 5.1 +
- Xcode 4.5 + Objective C 2.0
- Clang Compiler 4.2 (support for ObjC Literals)
- Objective-C Blocks (multithreading & callbacks)
- Unit Testing: OCMock & Expecta framework
- ARC (Automatic Reference Counting)
- Core Data with SQLite database and Model (database)

Push Notifications

- Device sends a unique push notification token to the web service.
- The service sends a push notification to Apple's or Google's push servers
- The push servers send the messages to the devices

All communications are being done using SSL-TLS security (built-in)

Push Notifications

- Device sends a unique push notification token to the web service.
- The service sends a push notification to Apple's or Google's push servers
- The push servers send the messages to the devices

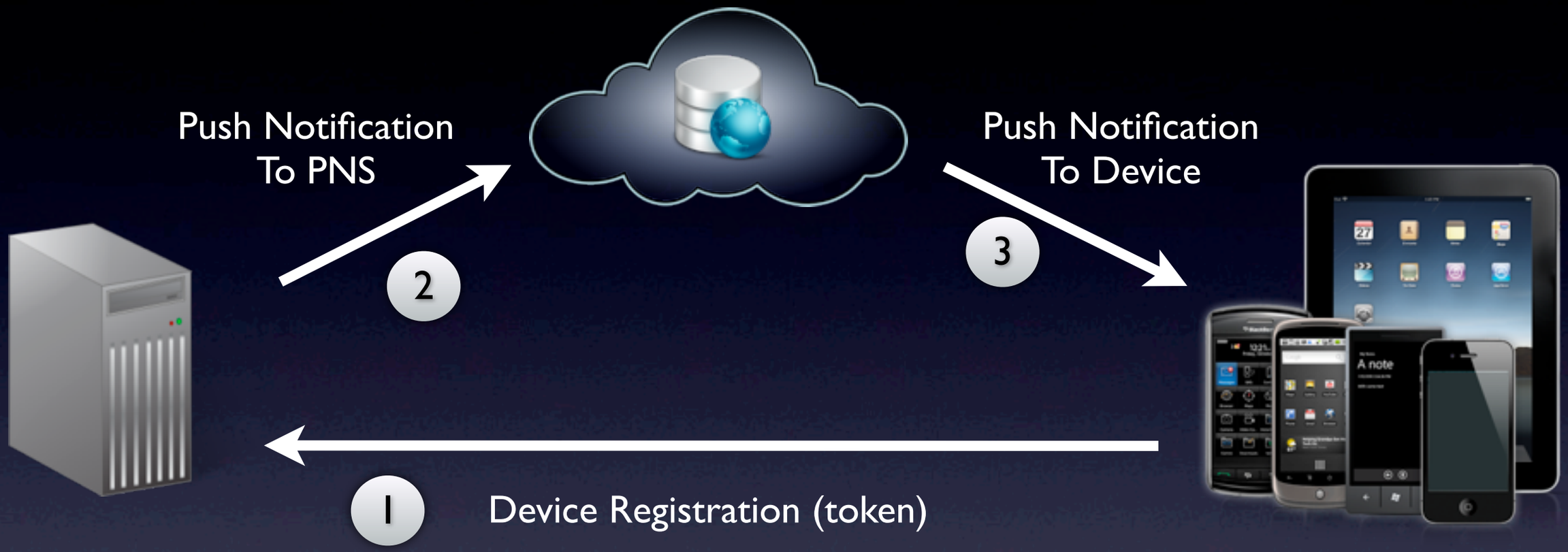
All communications are being done using SSL-TLS security (built-in)

Push Notifications

- Device sends a unique push notification token to the web service.
- The service sends a push notification to Apple's or Google's push servers
- The push servers send the messages to the devices

All communications are being done using SSL-TLS security (built-in)

Push Notifications



- Device sends a unique push notification token to the web service.
- The service sends a push notification to Apple's or Google's push servers
- The push servers send the messages to the devices

All communications are being done using SSL-TLS security (built-in)

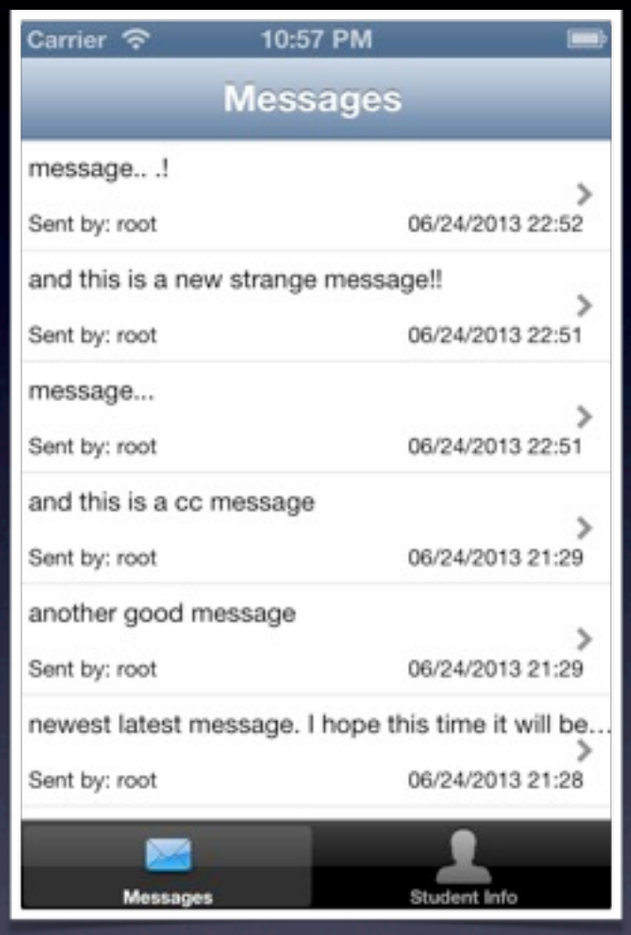
Functionality & API

Registration

- Application starts for the first time, checks registration status
- Sends Registration Request to the web server (push notification token, e-mail, name, etc)
- Server sends e-mail to the provided e-mail, including a confirmation link
- User clicks the link, and the server completes registration

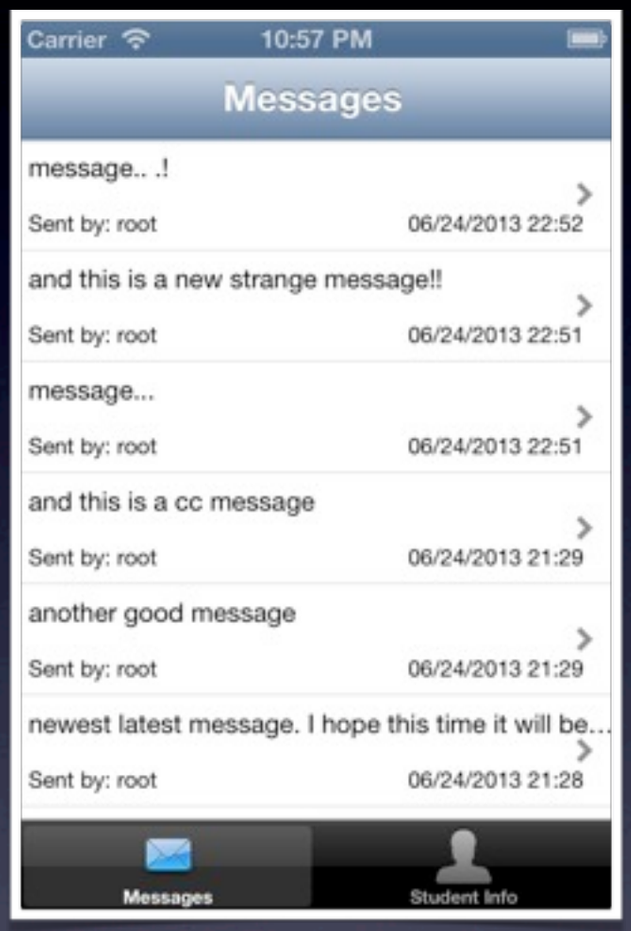
iOS Application

iOS Application

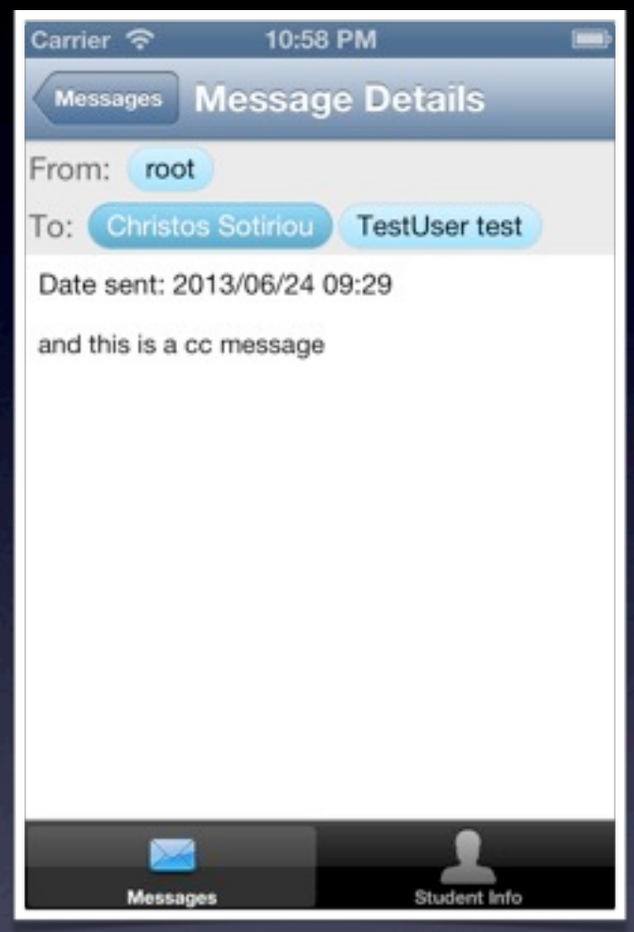


Read received messages

iOS Application

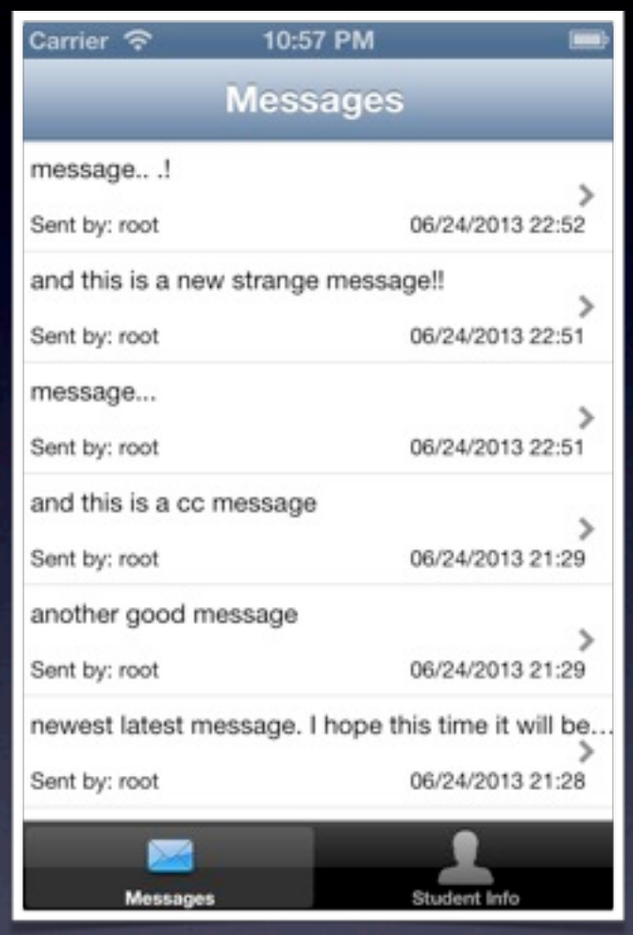


Read received messages

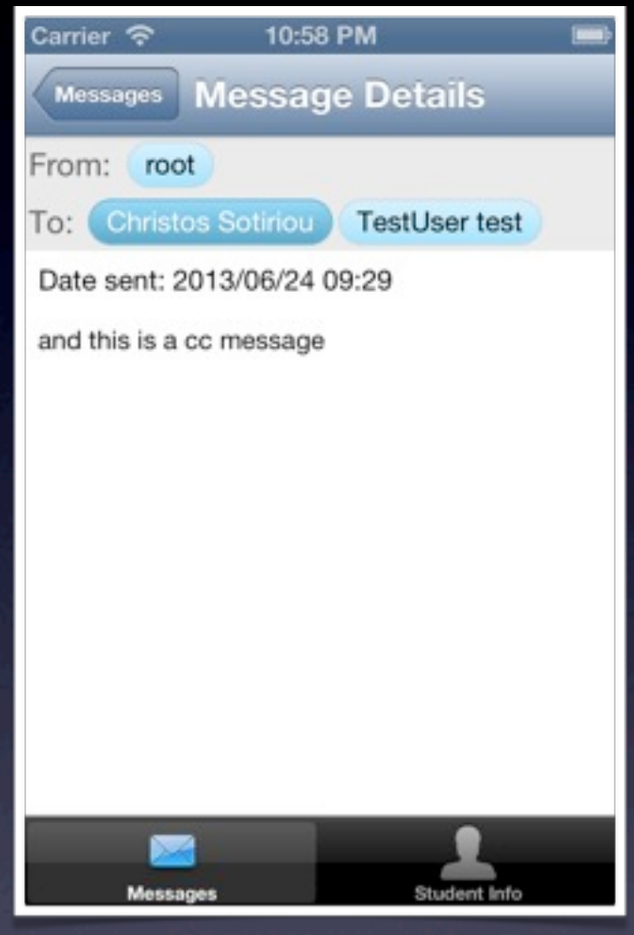


See message Details (cc info)

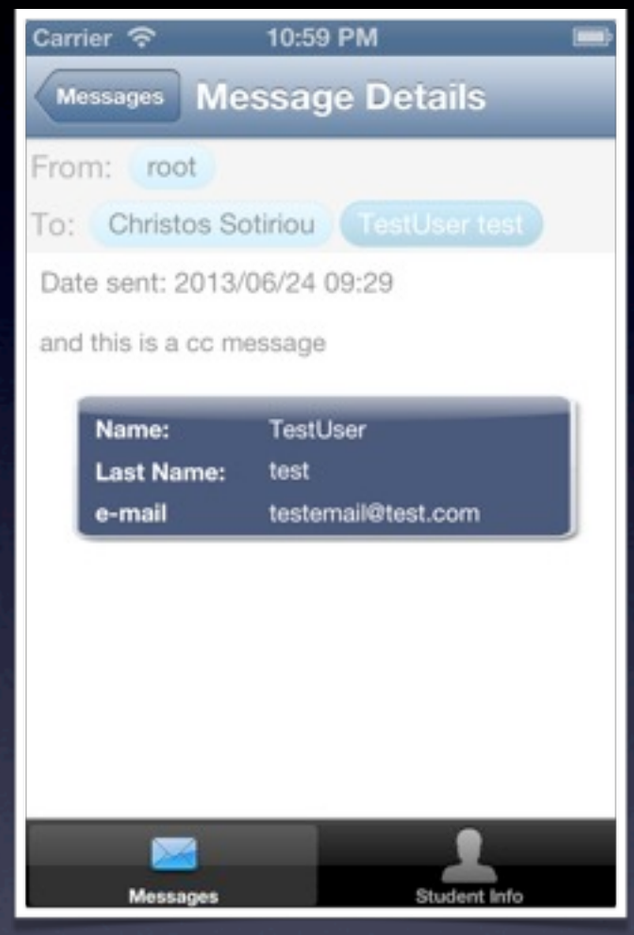
iOS Application



Read received messages



See message Details (cc info)



Click on a user name for details

Sending Messages

NotCenter Control Panel

Panel | Messages

Messages were sent successfully

Select	Name	LastName	Edit Student
☑	Christos	Sotiriou	Edit
☐	TestUser	test	Edit

Send message to selected students

kkk Send

 Logout

Sending Messages

NotCenter Control Panel

Panel | Messages

Messages were sent successfully

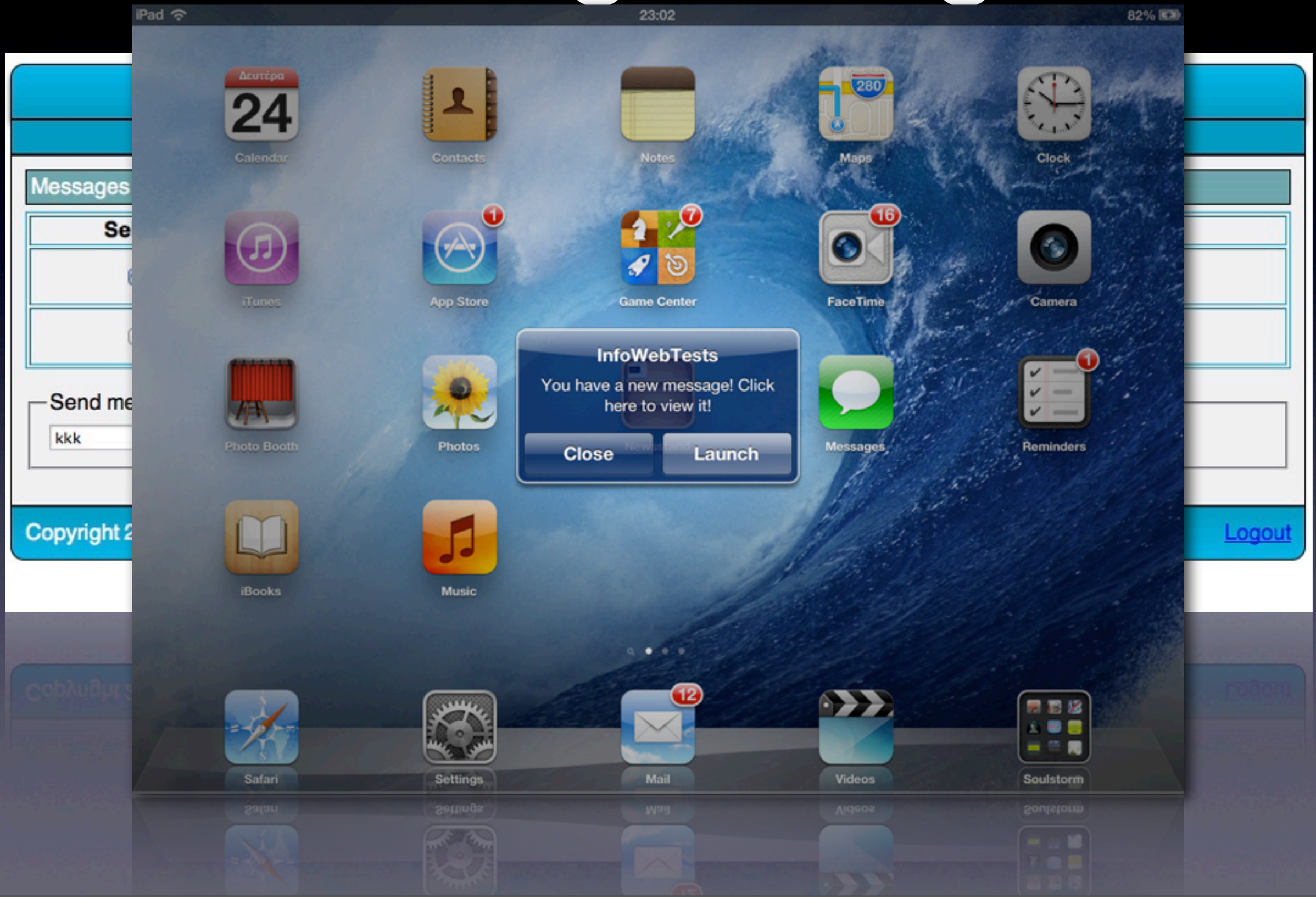
Select	Name	LastName	Edit Student
☑	Christos	Sotiriou	Edit
☐	TestUser	test	Edit

Send message to selected students

kkk Send

Copyright 2013 Telecom SudParis Logout

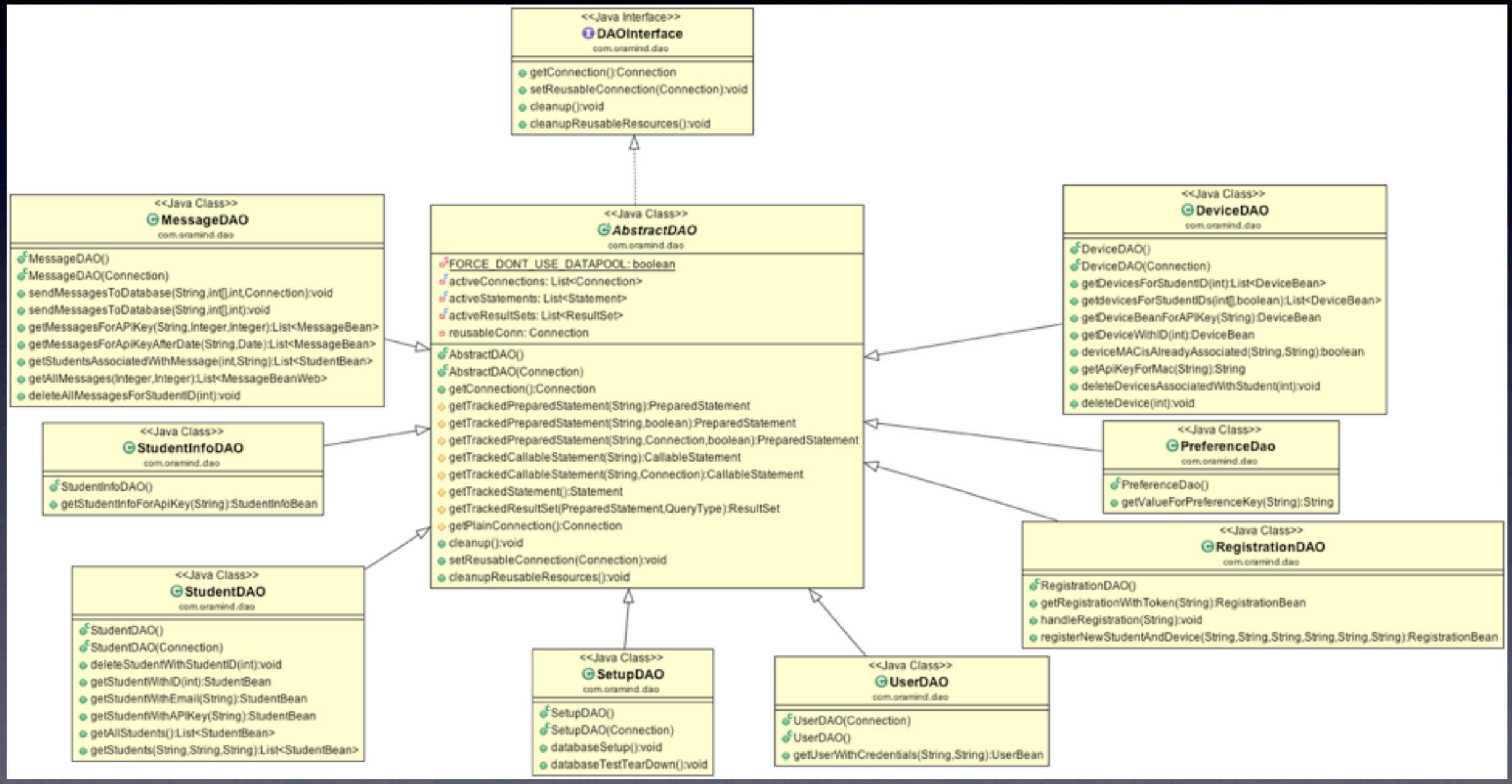
Sending Messages



Networking

Networking - Server

Networking - Server



Networking - Server

Networking - Server

- Abstract DAO (Data Access Object) class implements all networking facilities

Networking - Server

- Abstract DAO (Data Access Object) class implements all networking facilities
- Reusable Connections

Networking - Server

- Abstract DAO (Data Access Object) class implements all networking facilities
- Reusable Connections
- Multiple DAOs can share a common connection

Networking - Server

- Abstract DAO (Data Access Object) class implements all networking facilities
- Reusable Connections
- Multiple DAOs can share a common connection
- New connections from Data Pool (optional)

Networking - Server

- Abstract DAO (Data Access Object) class implements all networking facilities
- Reusable Connections
- Multiple DAOs can share a common connection
- New connections from Data Pool (optional)
- Single cleanup() function cleans up open connections in each DAO

Networking - iOS

Networking - iOS

- HTTP Client (queue-based, multithreaded singleton) - Grand Central Dispatch

Networking - iOS

- HTTP Client (queue-based, multithreaded singleton) - Grand Central Dispatch
- Implements block-based callbacks

Networking - iOS

- HTTP Client (queue-based, multithreaded singleton) - Grand Central Dispatch
- Implements block-based callbacks
- Holds API key (for requests)

Networking - iOS

- HTTP Client (queue-based, multithreaded singleton) - Grand Central Dispatch
- Implements block-based callbacks
- Holds API key (for requests)
- Asynchronous connections (up to 5 concurrent - can easily increase limit)

Networking - iOS

- HTTP Client (queue-based, multithreaded singleton) - Grand Central Dispatch
- Implements block-based callbacks
- Holds API key (for requests)
- Asynchronous connections (up to 5 concurrent - can easily increase limit)
- Can also be instantiated as simple object (for unit testing)

Testing

Testing

Testing

- Implemented a “Test Servlet”

Testing

- Implemented a “Test Servlet”
- Test capabilities can be enabled/disabled from the project’s config file (“DB.properties”)

Testing

- Implemented a “Test Servlet”
- Test capabilities can be enabled/disabled from the project’s config file (“DB.properties”)
- Filling the database with test data (manually)

Testing

- Implemented a “Test Servlet”
- Test capabilities can be enabled/disabled from the project’s config file (“DB.properties”)
- Filling the database with test data (manually)
- Testing performed by the iOS application

Testing

- Implemented a “Test Servlet”
- Test capabilities can be enabled/disabled from the project’s config file (“DB.properties”)
- Filling the database with test data (manually)
- Testing performed by the iOS application
- Stress Tests that create many concurrent sessions (up to 100)

Testing

- Implemented a “Test Servlet”
- Test capabilities can be enabled/disabled from the project’s config file (“DB.properties”)
- Filling the database with test data (manually)
- Testing performed by the iOS application
- Stress Tests that create many concurrent sessions (up to 100)
- Validation tests on the test data returned

Testing

- Implemented a “Test Servlet”
- Test capabilities can be enabled/disabled from the project’s config file (“DB.properties”)
- Filling the database with test data (manually)
- Testing performed by the iOS application
- Stress Tests that create many concurrent sessions (up to 100)
- Validation tests on the test data returned
- Test Servlet deletes the test data (manually)

Deliverables

- Web Service, Tomcat oriented
- Prototype iOS application, with all supported functionality implemented (+ tests)
- Installation documentation + Project Report (PDF)
- https://bitbucket.org/csotiriou/sai-csc7325-project/

The end

- https://bitbucket.org/csotiriou/sai-csc7325-project/