Overview of the Secure Notifications Tool

Decentralised Citizens ENgagement Technologies

Specific Targeted Research Project Collective Awareness Platforms

d-cent

FP7 – CAPS
Project no. 610349
D-CENT
Decentralised Citizens
ENgagement Technologies

Lead beneficiary: Thoughtworks

D5.6 Overview of the Secure Notifications Tool

October 2015
Version Number: 1

Authors:
John Cowie

Editors and reviewers:
Jaakko Korhonen
Pablo Aragón

ThoughtWorks®

The work leading to this publication has received funding from the European Union's Seventh Framework Programme (FP7/2007 – 2013) under grant agreement n° 610349.

Contents

Executive Summary

The Secure Notifications tool is a constituent piece of the development of task T5.1, a task to develop a suite of tools that enable digital democratic functions such as collaborative policy making, large-scale deliberation and voting.

The motivation for the development of a secure notifications tool as part of the D-CENT platform is to allow users to quickly view activities taking place on other parts of the platform. These activities would include actions such as votes, transactions, correspondence, or edits to documents. The hypothesis is that delivering these notifications to users will increase participation with the platform by highlighting opportunities to engage, respond and collaborate.

The tool has been designed to incorporate open standards that will allow easy integration of other existing and future tools, both inside and outside of the D-CENT platform. It has also been designed to support the verification of the integrity (and therefore authenticity) of notifications received, through cryptographic means.

User discovery and messaging has not been developed as part of the notifications application. Well maintained, existing open-source messaging solutions (such as Mattermost [1]) are available. It is felt that, where possible, the use and promotion of existing open source solutions is more responsible than building duplicates.

In addition to the Secure Notifications tool, two other applications were developed to support the integration of this tool into the D-CENT platform. The first application is a notifications server that can be deployed alongside other applications in the platform to assist with the storage and publishing of notifications. The second application is an adapter to convert open data from the City of Helsinki into a supported notifications format.

Application Repositories

All the software developed under the D-CENT umbrella is open-source. All of the code is developed under version-control using Git [2], and hosted on the github site. Details of each code repository is listed below.

Mooncake

Mooncake is the project title for the secure notifications tool, and will be the term used to refer to the tool for the remainder of this document.

The code is available at: https://github.com/ThoughtWorksInc/mooncakeA demonstration version of the app is hosted at: https://mooncake.dcentproject.eu

Coracle

Coracle is the project title for the notifications server.

The code is available at: https://github.com/ThoughtWorksInc/coracle

HelsinkiActivityStreams

The HelsinkiActivityStreams project is the adapter to transform data published by Helsinki City Council into notifications that can be consumed by Mooncake.

The code is available at: https://github.com/ThoughtWorksInc/HelsinkiActivityStream

Open Standards

Mooncake implements the following open standards:

Activity Streams 2.0

The Activity Streams 2.0 [3] specification () is currently being defined by W3C [4] as a standard for providing semantic descriptions of user actions on social media platforms. It incorporates the JSON-LD (JSON for Linking Data) format. Activity Streams 2.0 was chosen as the data format for notifications in the D-CENT platform. Adoption of this standard in this piece of software creates the potential for promoting and refining the standard to encourage further use in the open source community, and therefore facilitate integration with future components that provide other digital social functions.

JSON Web Signatures (JWS)

JWS [5] is a standard for signing content. The output is a base64 encoded message separated by periods into 3 parts: a header, the body, and the signature. The signature is generated by applying an asymmetric hash algorithm to the body of the message, combined with the private key of the message publisher. The message publisher's corresponding public key is used by a consumer to verify the message signature.

OpenID Connect

The OpenID Connect [6] standard is an industry standard authentication protocol. It adds another layer to the OAuth2 authentication protocol. When successfully issuing an authorisation token to the client, the user's details are signed using the JWS standard discussed above.

Platform Integration

The open standards above are used by Mooncake to integrate with existing tools in the D-CENT platform as well as an external API. These are:

Stonecutter

Stonecutter [7] is the single sign-on application developed as part of T5.1 and documented in deliverable D5.4. Mooncake integrates with Stonecutter to authenticate users, using OpenID Connect.

Objective8

Objective8 [8] is the policy-drafting tool developed as part of T5.1 and documented in deliverable D5.3. Objective8 publishes notifications to Mooncake using the Activity Streams 2.0 and JWS standards.

OpenAhjo

The OpenAhjo API [9] is an API external to the D-CENT platform and maintained by the City of Helsinki. Notifications are published to Mooncake with the use of the HelsinkiActivityStreams adapter.

High Level Feature Overview

1. Viewing Activity Feed

Feature definition: Users can view a feed of activities from AS2 endpoints.

User Story:

As a Mooncake user

I want to view a feed of activities

So that I can discover activity taking place on other connected applications.

Description: Users are shown activities in a single feed in descending order chronological order. Activity icons are colour coded by the source they were retrieved from. The time the activity was published is converted into a human-friendly format (e.g. 10 minutes ago). Activity sources are described in a configuration file.

Technical Implementation: Activities are loaded from JSON retrieved over HTTP. The activity sources are configured in a YAML file with the URL of the published activity stream. For the implementation of this story, HTTP requests are made to the activity sources each time the feed page is loaded. Asynchronous activity loading was completed in a later story (see 3).

Screenshots:

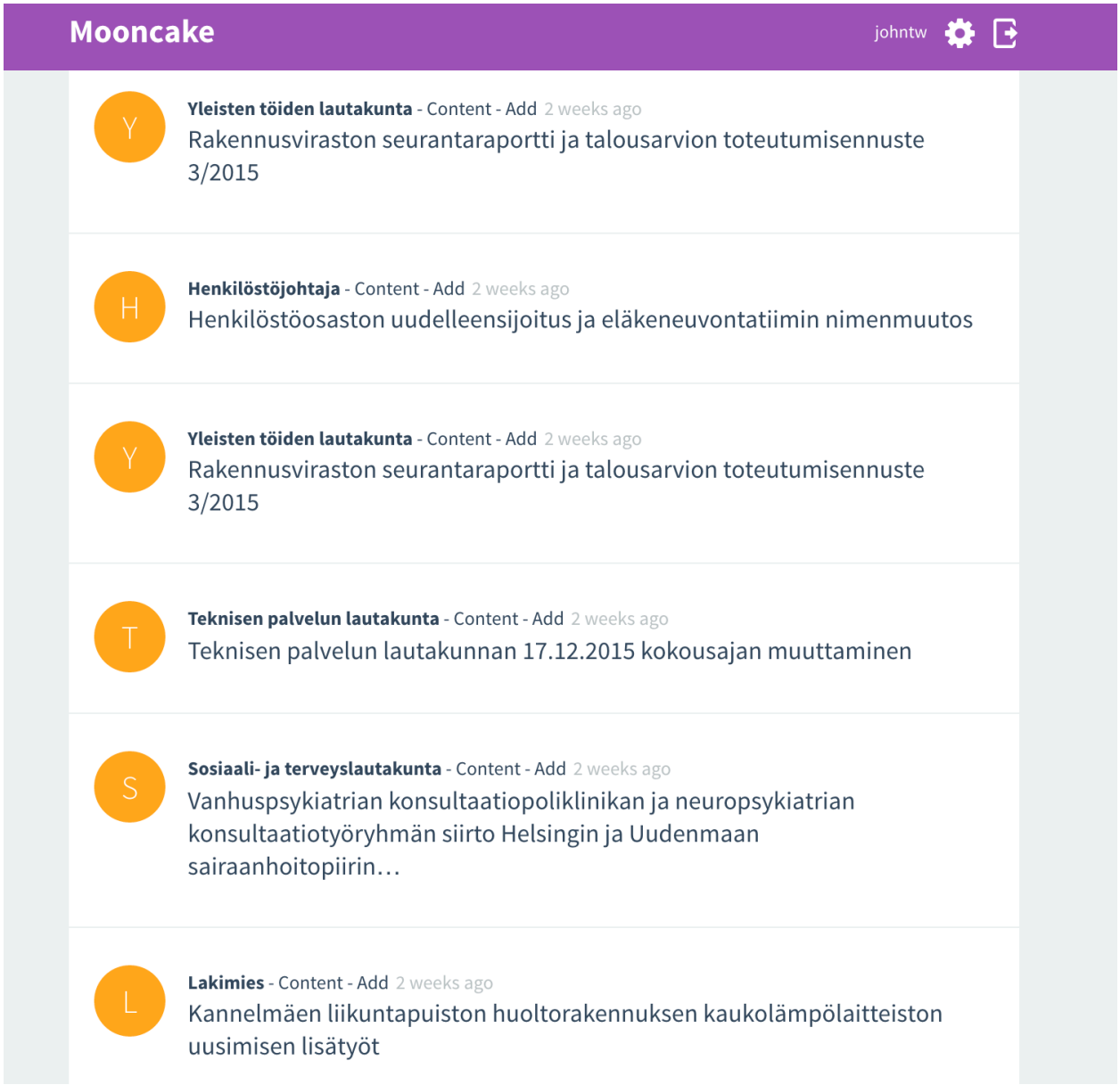

Figure 1:Screenshot of feed

2. User sign-in

Feature definition: Users can sign in to Mooncake with their Stonecutter (OAuth Server) account.

User Story:

As a Mooncake user

I want to sign in with Stonecutter

So that I can manage my feed

Description: If users are not signed in then they are taken to the login page and can click to sign-in with their D-CENT account. If signing in for the first time they are then asked to create a Mooncake account by choosing a username. Signed-in users are able to sign out of their account by clicking the sign-out icon, and their session is terminated.

Technical Implementation: Mooncake integrates with Stonecutter using the OpenID Connect protocol. This interaction is described in detail in deliverable D5.4. Once a user is successfully signed in and has created an account, their account details are saved in MongoDB.

Screenshots:

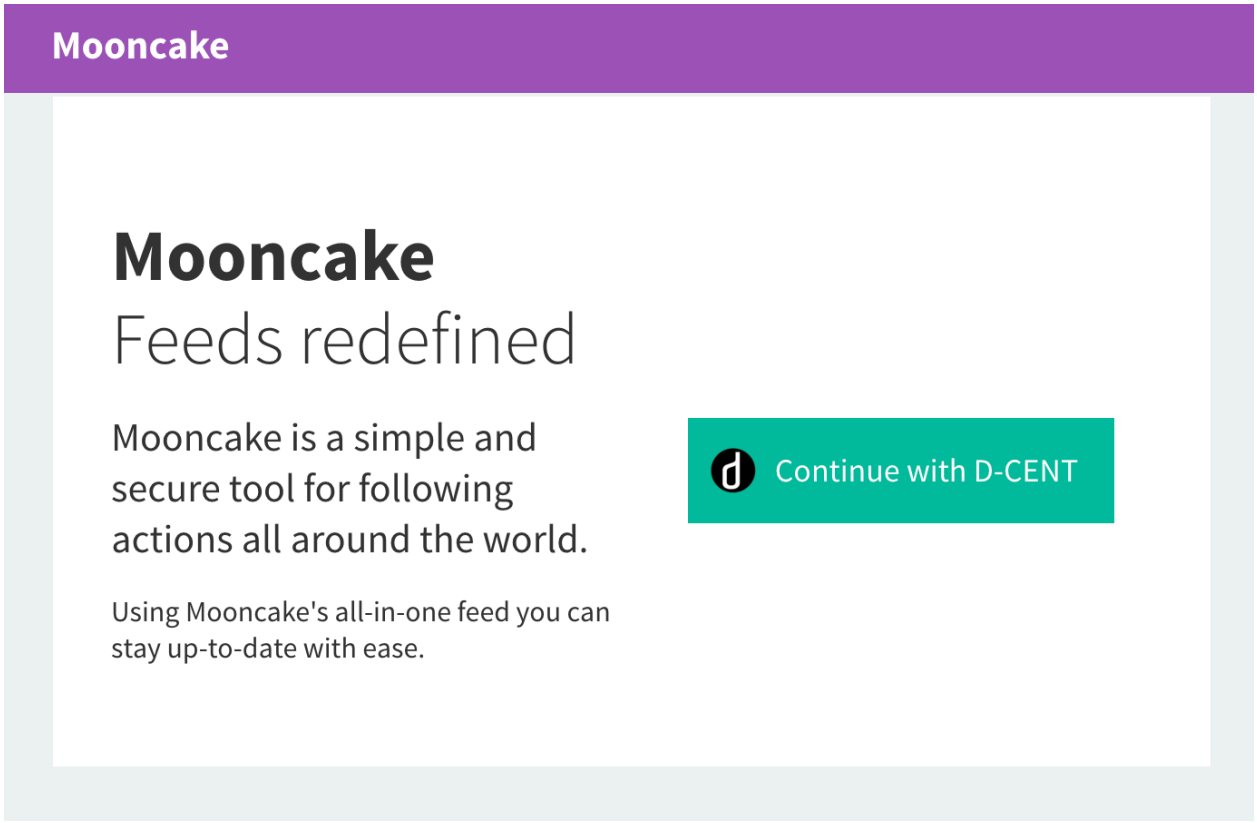

Figure 2: Screenshot of Mooncake sign-in page

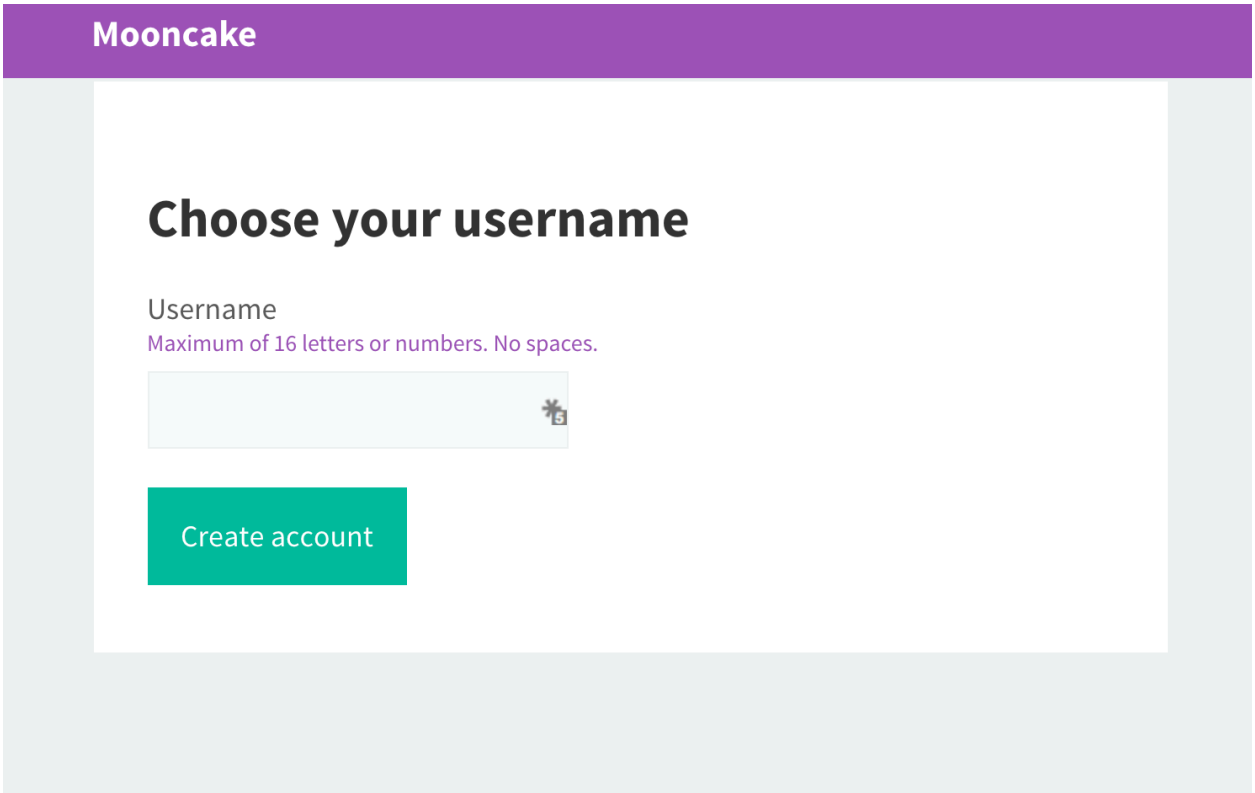

Figure 3: Screenshot of account creation page

3. Asynchronous Feed Updates

Feature definition: Activity sources are polled periodically to retrieve new activities.

User Story:

As an activity stream publisher

I don't want an HTTP request to be made to my activity stream each time a feed page is loaded

So that the load on my server is reduced

Description:

Activities are requested from the configured Activity Stream sources periodically. The time period can be configured when the application is deployed. When the feed is requested by a user, the activities are loaded from the database.

Technical Implementation: A thread is created when the application starts to asynchronously poll the activity stream source URLs. When the activities are loaded they are stored in MongoDB. The timestamp of the most recently stored activity is also stored for each source. Each subsequent request to the activity stream sources uses this stored timestamp to query the API for new activities.

4. Objective8 Integration

Feature definition: When users perform activities in Objective8, notifications of these activities are displayed in the Mooncake feed.

User Story:

As a user of the D-CENT platform

I want to receive notifications in Mooncake when actions are performed in Objective8

So that I can quickly get up-to-date with the status of the policy drafts.

Description: Objective8 publishes an activity stream JSON API with activities for the creation of objectives and the addition of questions to the objectives.

Technical Implementation: For this story an activity stream utility (Coracle) was created to allow applications to post activities and surface a queryable API endpoint. This Activity Stream is deployed alongside Objective8 and stores activities into MongoDB. Objective8 was updated to post activities to this utility when objectives and questions are created by users.

Screenshots:


```
[
  - {
      - object: {
          - object: {
                displayName: "CONNETTIVITA' GRATUITA",
                @type: "Objective"
            },
            url: "http://objective8.dcentproject.eu/objectives/20/questions/21",
            displayName: "prendiamo come esempio la legge svedese? o altre leggi simili nel mondo?",
            @type: "Objective Question"
        },
      - actor: {
            displayName: "fabrisspec",
            @type: "Person"
        },
        published: "2015-10-22T16:59:54.363Z",
        @type: "Question",
        @context: "http://www.w3.org/ns/activitystreams"
    },
  - {
      - object: {
            url: "http://objective8.dcentproject.eu/objectives/20",
            content: "Test per la proposta di legge per la realizzazione della connettività gratuita a tutti i cittadini italiani",
            displayName: "CONNETTIVITA' GRATUITA",
            @type: "Objective"
        },
      - actor: {
            displayName: "pandeussilvae",
            @type: "Person"
        },
        published: "2015-10-22T16:23:54.099Z",
        @type: "Create",
        @context: "http://www.w3.org/ns/activitystreams"
    },
```


Figure 4: Screenshot of results from the Objective8 Activity Stream API

5. OpenAhjo Integration

Feature definition: When Helsinki City Council publishes information about council decisions notifications of these decisions are displayed in the feed.

User Story:

As a Helsinki Citizen

I want to receive notifications in Mooncake when decisions are added to the OpenAhjo API

So that I can keep up-to-date with decisions that my city council are making

Description: The Helsinki Activity Streams utility was created to poll the OpenAhjo API for decision information and then store these activities in a database and surface them with a queryable API that is compatible with Mooncake. The OpenAhjo API does not use the Activity Streams 2.0 format, so the utility also maps the data from the original format to AS2.

Technical Implementation: The mapping utility makes periodic asynchronous HTTP requests to the Open Ahjo API, converts the results to the AS2 format and then pushes them to an instance of Coracle (where they are stored in MongoDB).

Screenshots:

[
- {
"@context": "http://www.w3.org/ns/activitystreams",
"@type": "Add",
- actor: {
"displayName": "Asuntolautakunta",
"@type": "Group",
"@id": "http://dev.hel.fi/paatokset/v1/policymaker/29/"
},
"published": "2015-05-10T16:43:21.000Z",
- object: {
"displayName": "Tutustumismatka valtakunnallisille asuntomessuille",
"content": " Lautakunta päätti tehdä tutustumismatkan Tampereella pidettäville valtakunnallisille asuntomessuille 7.8.2012. Samalla l
matkakustannuksista, sisäänpääsymaksuista ja mahdollisista päivärahoista aiheutuvat KVTES:in mukaiset kustannukset maksetaan asuntolautakunn
"@type": "Content",
"@id": "http://dev.hel.fi/paatokset/v1/agenda_item/33236/",
"url": "http://dev.hel.fi/paatokset/asia/hel-2012-008324/aslk-2012-8/"
},
- target: {
"displayName": "Tutustumismatka valtakunnallisille asuntomessuille",
"content": "Tampereen Vuoreksessa sijaitsevan asuntomessualueen ja koko uuden kaupunginosan lähtökohtana on alunperin Englannissa syntynyt, mu
puutarhakaupunkimalli. Vuoreksesta on matkaa Tampereen keskustaan noin seitsemän kilometriä.",
"@type": "Content",
"@id": "http://dev.hel.fi/paatokset/v1/issue/26/"
}
},
- {
"@context": "http://www.w3.org/ns/activitystreams",
"@type": "Add",
- actor: {
"displayName": "Asuntolautakunta",
"@type": "Group",
"@id": "http://dev.hel.fi/paatokset/v1/policymaker/29/"
},
"published": "2015-05-10T16:43:21.000Z",
- object: {
"displayName": "Lausunnon antaminen Asumisen rahoitus- ja kehittämiskeskukselle Helsingin kaupungin asunnot Oy:n korkotukihakemuksista",
"content": " Asuntolautakunta päätti lausuntonaan Asumisen rahoitus- ja kehittämiskeskukselle puoltaa seuraavia vuokratalojen rakentamisen k
<table class="msoUcTable" tabindex="-1" border="1" bordercolor="buttontext"> <colgroup> <col> <col> <col> </colgroup> <tbody valign="top"> <
 </td> <td> Taiteilija-asunnot </td> <td> 14 asuntoa </td> </tr> <tr> <td> Heka Viikinmäki </
Lättiläisenkatu 12 </td> <td> 50 asuntoa </td> </tr> <tr> <td> Heka Vallila </td> <td> Kangasalanti
56 asuntoa </td> </tr> <tr> <td> Yhteensä </td> <td>   </td> <td> 120 asuntoa </td> </tr> </tbo

Figure 5: Screenshot of results from the HelsinkiActivityStream API

6. Feed customisation

Feature definition: Users can configure which types of activities are displayed in their feed.

User Story:

As a Mooncake user

I want to be able to configure which activities are shown

So that I only view the types of activities that I am interested in.

Description: Users can click the settings icon to be taken to a page where they can select or deselect which types of activities are shown in their feed. If JavaScript is enabled then users are also able to select or deselect all types for a given activity stream source.

Technical Implementation: The users preferences are sent with an HTTP post to the backend, and stored in their account record in MongoDB. When retrieving the activities to display in the feed, the user's preferences are used to construct the query and select the required activities from the database.

Screenshots:

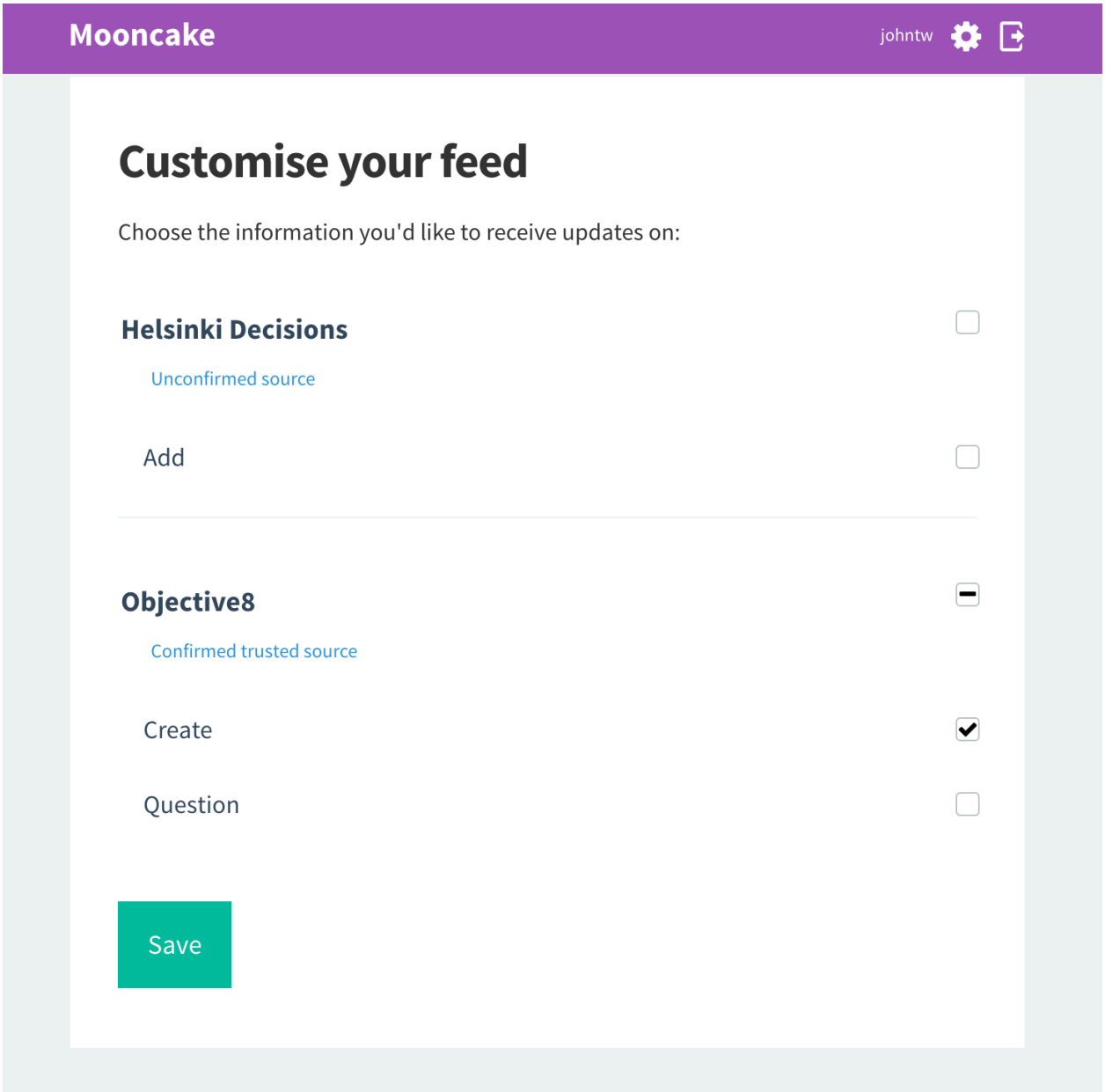

Figure 6: Screenshot of feed customisation page

7. Feed pagination

Feature definition: The user is only shown the most recent results in the feed, and can then navigate between pages of results to view older activities.

User Story:

As a Mooncake user

I only want to view older results if I choose to

So that the feed loads more quickly

Description: The most recent 50 activities are shown to the user on the feed page. At the bottom of the feed a button is presented to the user to navigate to a page of the 50 preceding activities. On pages other than the first and last, the user is presented with buttons to navigate forwards and backwards in time. On the last page the user is only shown a button to go forward to newer results.

Technical Implementation: The feed pages are retrieved by supplying a page-number query parameter to the server side. The required results are retrieved from the database using MongoDB querying syntax to order and chunk the activities. If the query parameter supplied is invalid the user is presented with a 404 page.

Screenshots:

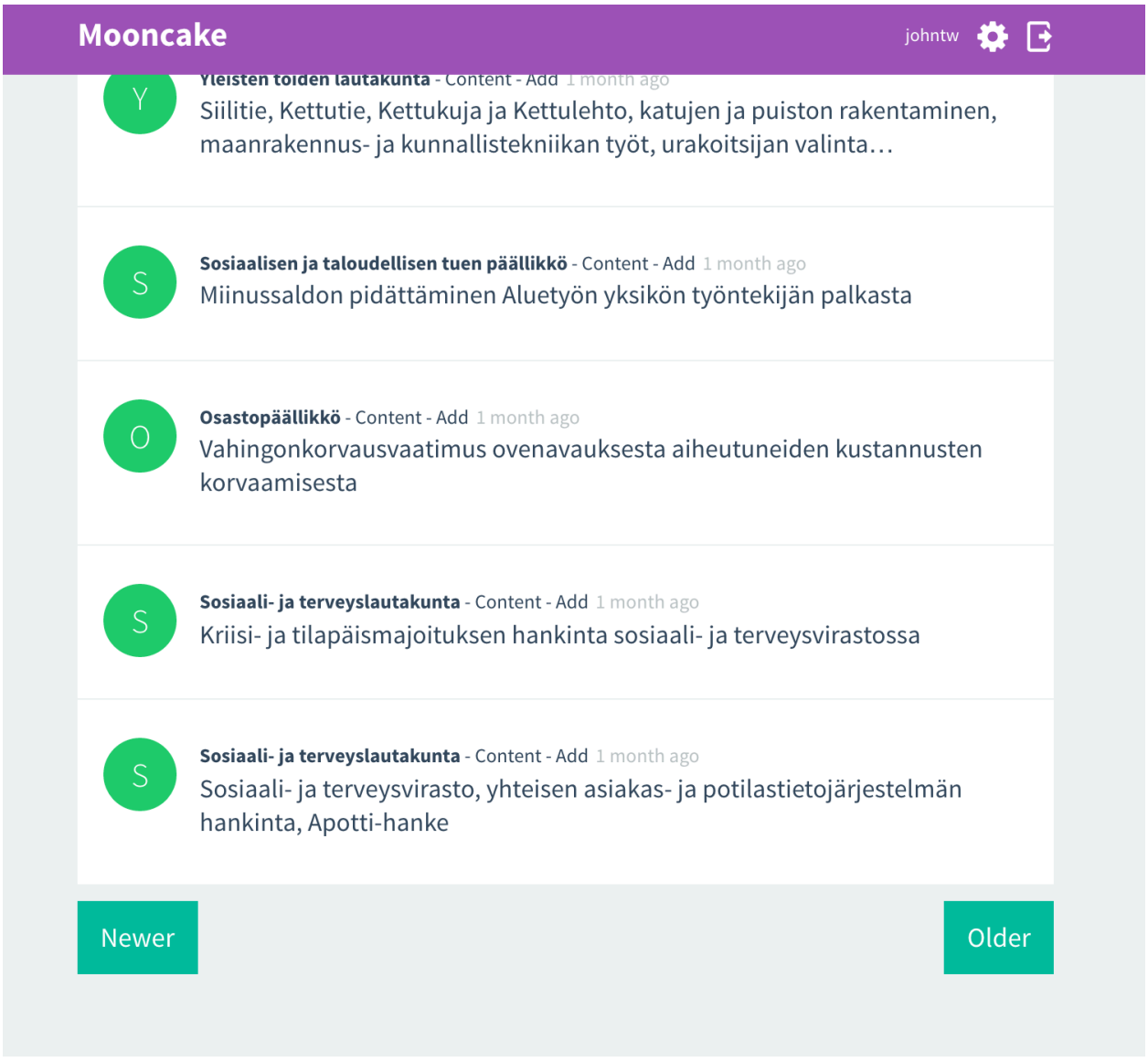

Figure 7: Screenshot of feed with pagination buttons

8. Activity Stream signing

Feature definition: Activities are signed using the JWS (JSON Web Signature) standard and their signatures are verified by Mooncake.

User Story:

As a user of Mooncake

I want activities to be signed in transit between Activity Stream sources and Mooncake

So that I can be sure of the integrity of the notifications.

Description: Coracle supports the signing of the results exposed in the API. Mooncake verifies the signature when retrieving the activities, and stores the status of the signature (i.e. signed or unsigned) when storing the activities. A visual cue is added to messages in the feed that are unsigned or were unsuccessfully verified, along with an explanation, to indicate to the user that the integrity of the notification is not assured.

Technical Implementation: Coracle was updated to encode and sign the messages using the JWS standard. The jose4j Java library [10] was used to create the signatures. Coracle also generates a public/private key pair on startup, and exposes the public key at another endpoint that is linked to in the JSON result. Mooncake uses the jose4j library to decode the result and verify the signature. The status of the verification is stored in the database in the activity document. This status is checked to determine whether to render a warning to the user.

Screenshots:

Figure 8: Screenshot of JWS signed activities

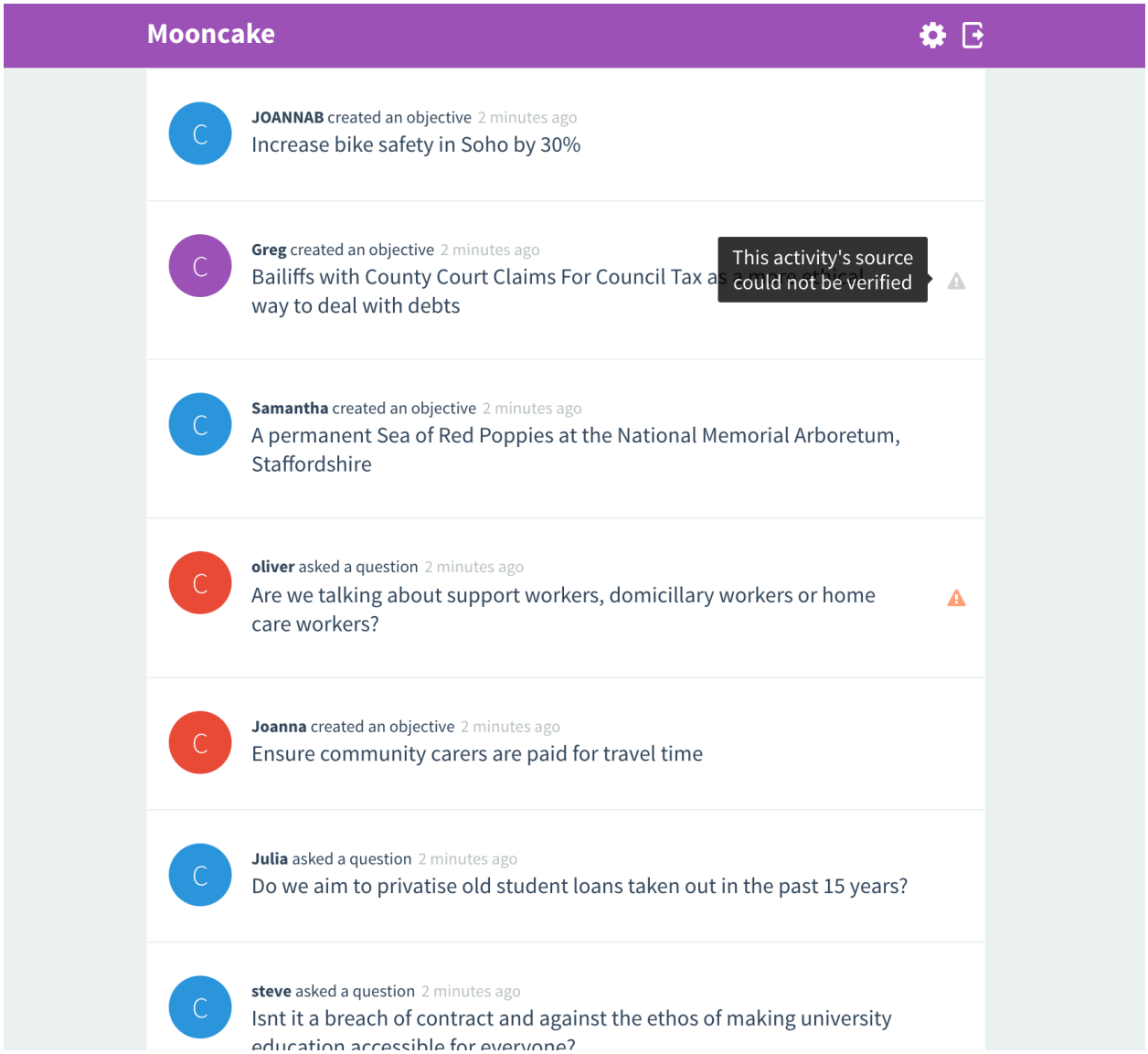

Figure 9: Screenshot of activities with warnings indicating signature verification failure

Architectural Design

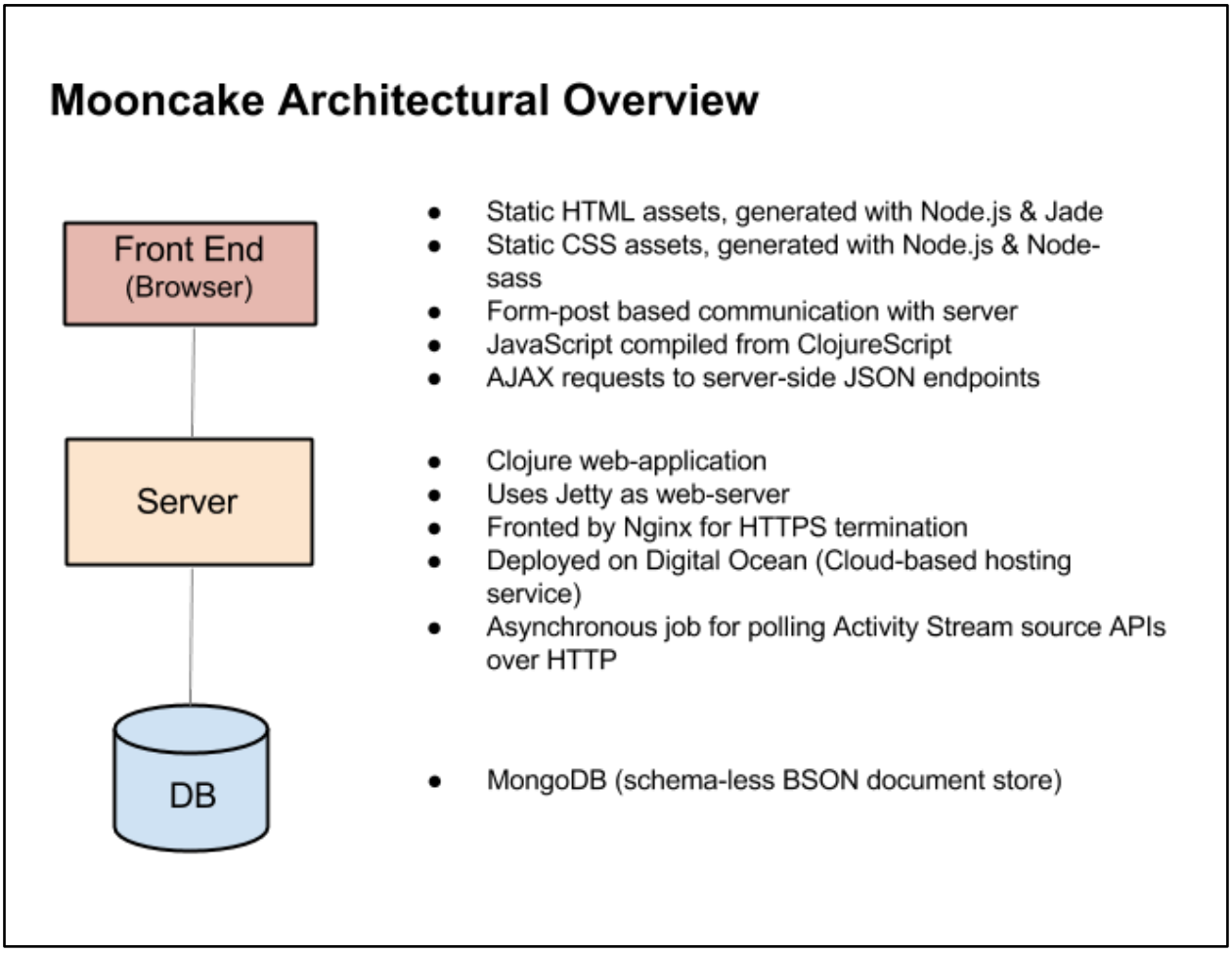

Figure 10: Application architecture

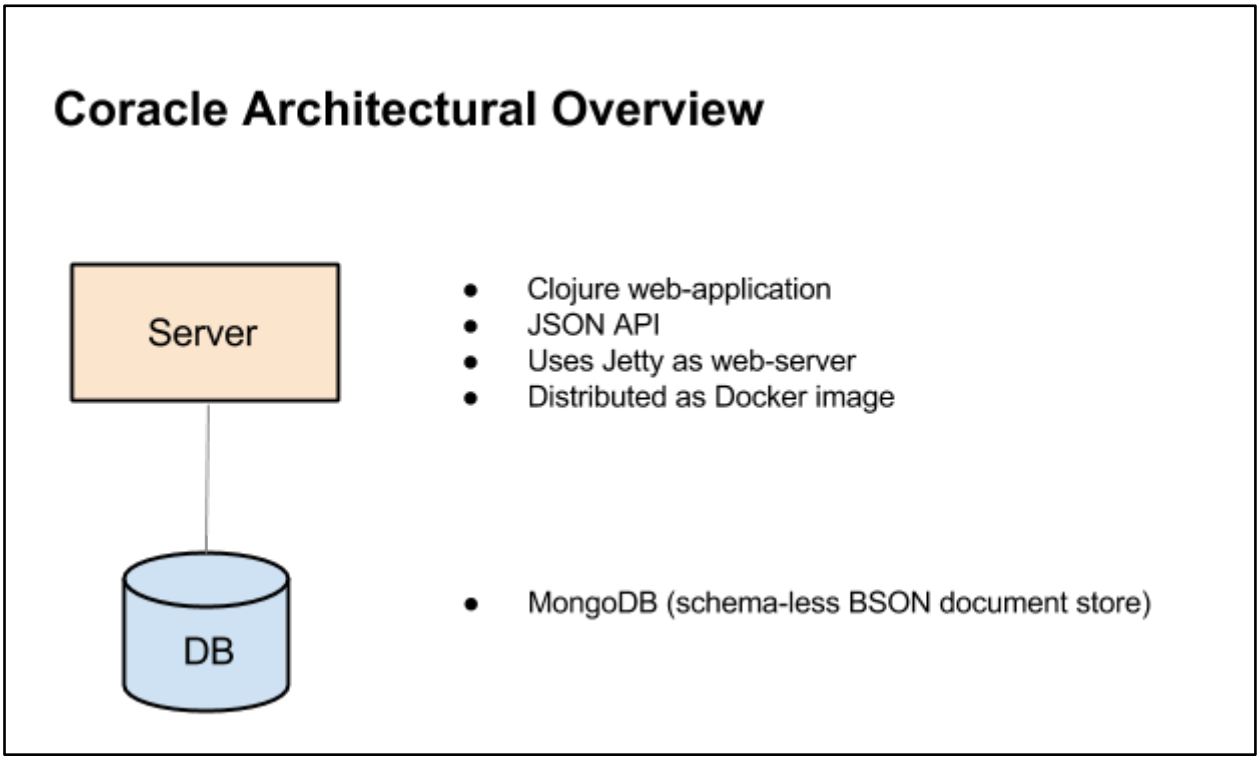

Figure 11: Notifications server architecture

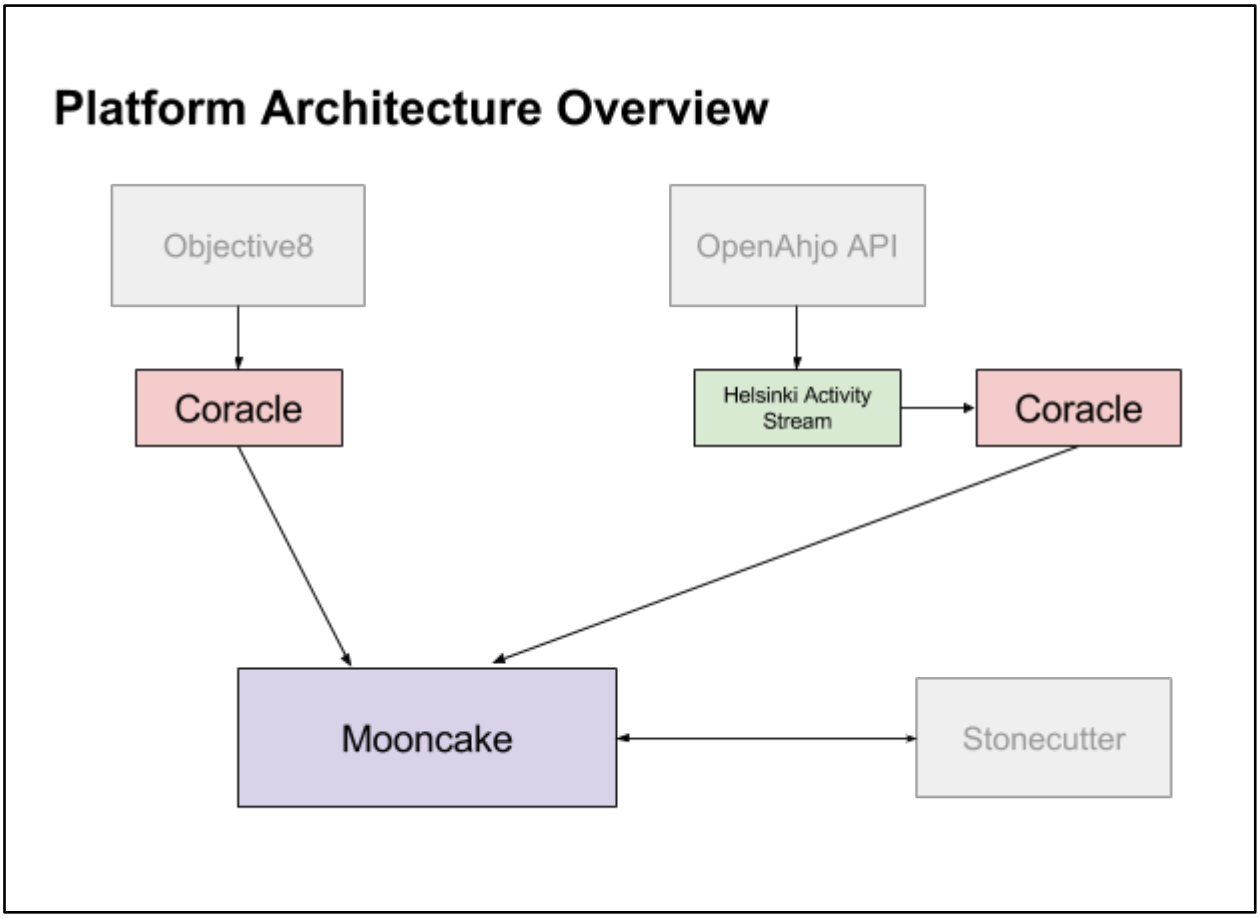

Figure 12: Platform architecture overview

Usage of Open Standards

ActivityStreams 2.0

Some initial AS2 types have defined for both Objective8 and the OpenAhjo API as a proof of concept. Future work will include defining vocabulary for a greater number of types, so that a larger range of actions in the D-CENT platform can produce notifications that are displayed in Mooncake.

Objective8 Types

Objective Creation:

```
{
  object: {
            url: <objective-url>,
            content: <objective-content>,
            displayName: <objective-title>,
            @type: "Objective"
  },
  actor: {
            displayName: <username>,
            @type: "Person"
  },
  published: <published-timestamp>,
  @type: "Create",
  @context: "http://www.w3.org/ns/activitystreams"
}
```

Question Addition

```
{
  object: {
            object: {
                        displayName: <related-objective-title>,
                        @type: "Objective"
            },
            url: <question-url>,
            displayName: <question-content>,
            @type: "Objective Question"
  },
  actor: {
            displayName: <username>,
            @type: "Person"
  },
  published: <published-timestamp>,
  @type: "Question",
  @context: "http://www.w3.org/ns/activitystreams"
}
```

OpenAhjo Types

Decision

```
{
  actor:  {
            displayName: <decision-making-group>,
            @type: "Group",
            @id: <group-url>
  },
  object: {
            displayName: <decision-title>,
            content: <decision-content,
            @type: "Content",
            @id: <issue-url>,
            url: <decision-url>
  },
  target: { displayName: <parent-issue-display-name>,
            content: <parent-issue-description>,
            @type: "Content",
            @id: <issue-id>
  },
  published: <published-timestamp>,
  @type: "Add",
  @context: "http://www.w3.org/ns/activitystreams",
}
```

JSON Web Signature (JWS)

The hash algorithm chosen for generating the signature for activities is RSASSA-PKCS-v1_5 using SHA-256 hash. This algorithm recommended by the JavaScript Object Signing and Encryption (JOSE) Working Group. The public key is published by Coracle as a JSON Web Key (JWK) [11].

OpenID Connect

Mooncake currently integrates with Stonecutter to authenticate users. The authentication flow between Stonecutter and client applications is documented in deliverable D5.4.

Technology Rationale

Clojure [12]:

- JVM-based language, allowing for familiar deployment
- Expressive dynamically-typed language with an emphasis on high-quality code principles such as immutability by default
- Strong and growing open-source community
- Easy Java interoperation, enabling use of large body of existing Java libraries
- Good level of experience with the ThoughtWorks team, allowing the team to deliver features quickly and at a high-level of quality

MongoDB [13]:

- High level of adoption in industry, so well supported
- Shallow initial learning curve
- Easy to deploy
- Designed for scalability and resiliency
- JSON document store makes it a convenient choice for storing JSON activities

Sass [14]:

- More powerful syntax for producing CSS
- Allows removal of duplication in CSS with variables

ClojureScript [15]:

- Performance of JavaScript improved with Clojure's persistent data structures
- Allows cross-compilation of Clojure code to both client and server-side
- Emphasis on immutability leads to simpler, more robust client-side code

Python [16] (Helsinki Activity Stream mapper):

- High level of adoption with developers in Helsinki

Security

All the applications are deployed with TLS certificates (HTTPS) to ensure encryption of traffic between nodes and between servers and clients (browsers).

Activities are signed to ensure data-integrity (see preceding section on JWS).

Stonecutter is used for authenticating users to Mooncake. Delegating user sign-in to Stonecutter ensures that a minimal amount of personal user data is stored in Mooncake.

The OWasp Top Ten security [17] guidelines were reviewed and followed. In particular Cross-Site Request Forgery was mitigated with the use of session-based CSRF tokens.

Deployment

A demonstration version of Mooncake is deployed to a server located in London provided by DigitalOcean [18] (a cloud-hosting service). The app is deployed using a combination of Ansible [19] and Docker [20]. Ansible is an automated deployment and configuration tool, and is used to provision the Digital Ocean environment with application dependencies. Docker containers within the virtual environment are used to host the application and the database. Each time the application is updated a new version is uploaded to the server and deployed into a docker container. The Ansible provisioning scripts are included in the code repository. The tools in the D-CENT platform are designed to be self-hosted - organisations are able to use the deployment code provided to host the applications on a server of their choosing.

A Docker image that contains Coracle is uploaded to DockerHub [21] as part of the Coracle Continuous Integration pipeline. DockerHub is a cloud-hosted repository for Docker images. The image can be pulled from DockerHub and run alongside other applications (such as Objective8) as part of the deployment process.

Database Design

MongoDB is a logical choice for storing activities, as it is schema-less and thus supports storing documents of different shapes (i.e. activities of different types) in the same collection. It also stores data in the BSON (Binary JSON) data format, which works well for JSON activities. Three different MongoDB collections are used in Mooncake and are described below.

Mongo Collections

activity

Activities retrieved from the AS2 sources are stored as documents in the activity collection, with the addition of an `activity-src` field that indicates which AS2 source the activity was retrieved from, and a `signed` field that indicates whether the signature of the activity was successfully verified.

activityMetadata

The activityMetadata collection stores details of the timestamp of the most recently retrieved activity for a given AS2 source, along with an enumeration of types that have so far been retrieved from the source.

The format of an activityMetadata document is:

```
{
  "_id" : <document-id>,
  "activity-src" : <activity-src-id>,
  "latest-activity-datetime" : <timestamp>,
  "activity-types" : <list-of-type-ids>
}
```

For example:

```
{
  "_id" : ObjectId("1234"),
  "activity-src" : "objective8",
  "latest-activity-datetime" : "2015-10-27T10:13:41.182Z",
  "activity-types" : [ "Create", "Question" ]
}
```

user

The user collection stores details of a user's mooncake account, along with their feed preferences.

The format of a user document is:

```
{
  "_id" : <document-id>,
  "auth-provider-user-id" : <stonecutter-account-id>,
  "username" : <mooncake-username>,
  "feed-settings" : <mapping-of-activity-src-to-type>
}
```

An example of a user document is:

```
{
  "_id" : ObjectId("1234")
  "auth-provider-user-id" : 1234,
  "username" : "userA",
  "feed-settings" : {
    "objective8" : {
      "types" : [ { "id" : "Create", "selected" : true },
                  { "id" : "Question", "selected" : true } ]
    },
    "OpenAhjo" : {
      "types" : [ { "id" : "Add", "selected" : true } ]
    }
}
```

Testing

The development team adopted test-driven development (TDD) as a standard practise. Unit tests were written before implementing the corresponding features to ensure the testability of the code and protect against regression of the feature set.

In addition higher level automated tests were written. These included both tests of the integration of different modules of the code, and end-to-end automated browser tests.

The team also practised Continuous Integration (CI) and Continuous Delivery (CD). SnapCI [22] is a cloud-hosted tool used by the team for CI. Each time a commit is pushed to the code repository, the full automated test suite is run. If the tests pass, the application is then automatically redeployed to a staging environment where the application can be manually tested for further quality assurance.

Features In Development

The following are details of Stonecutter features currently being analysed and developed:

JavaScript loading of older activities

Users will be able to load older notifications in their feed without refreshing the page. This will be done with AJAX requests to the server.

Mooncake 'locked-down' mode

Administrators of Mooncake will be able to configure the application to run in a locked-down mode where only users that have been marked as trusted by the administrator of Stonecutter will be able to access the application.

Activity Streams 2.0

The orderedCollection AS2 type for wrapping collections of activities will be used in the published ActivityStream APIs.

The palette of mappings between Objective8 actions and AS2 types will be extended.

Integration

AS2 types will be defined for Freecoin (another tool in the D-CENT platform, described by work package D5.5). Freecoin will be integrated with Mooncake by installing an instance of Coracle for alongside the application.

Future Work

Some future possible developments that have been identified are:

- Indexing the activities in a search tool (such as ElasticSearch [23]) to allow users to search through activities.
- Integration with Hackpad [24] and/or Objective8 in such a way that users are able to create a collaborative document to respond to notifications.
- Integration with an open-source messaging service such as Mattermost
- The launch of a pilot of Mooncake with users in Helsinki.

References

1. http://www.mattermost.org
2. https://github.com
3. http://www.w3.org/TR/activitystreams-core
4. http://www.w3.org
5. https://tools.ietf.org/html/draft-ietf-jose-json-web-signature-41
6. http://openid.net/connect
7. https://github.com/ThoughtWorksInc/stonecutter
8. https://github.com/ThoughtWorksInc/objective8
9. http://dev.hel.fi/apis/openahjo
10. https://bitbucket.org/b_c/jose4j/wiki/Home
11. http://tools.ietf.org/html/rfc7517
12. http://clojure.org
13. https://www.mongodb.org
14. http://sass-lang.com
15. https://github.com/clojure/clojurescript
16. https://www.python.org
17. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
18. https://www.digitalocean.com
19. http://www.ansible.com
20. https://www.docker.com
21. https://hub.docker.com/r/dcent/coracle
22. https://snap-ci.com
23. https://www.elastic.co
24. https://hackpad.com