INFRASTRUCTURE AND INTEGRATED TOOLS FOR PERSONALIZED LEARNING OF READING SKILL

D8.1– System Architecture

Document identifier	**iREAD_D8.1_SystemArchitecture.final.docx**
Date	**2017-07-17**
WP	**WP8**
Partners	**NTUA, ULBS**
WP Lead Partner	**NTUA**
Document status	**Final**

Grant Agreement number:
731724 — iRead H2020-ICT-2016-2017/H2020-ICT-2016-1

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No 731724

Deliverable Number	D8.1
Deliverable Title	System Architecture
Deliverable version number	V1.0
Work package	WP8
Task	Task 8.1 iRead System Architecture
Nature of the deliverable	Report (R)
Dissemination level	Public
Date of Version	2017-07-17

Author(s)	P. Georgantas
Contributor(s)	A.Symvonis, C. Raftopoulou, K.Karpouzis
Reviewer(s)	I.Mihu, C.Mihu, M.Karlsson
Abstract	This document presents the System Architecture of iRead Infrastructure, the Logical and Physical Architecture of the system, the software components and the APIs exposed to the clients.
Keywords	System requirements, software architecture

Document Status Sheet

Issue	Date	Comment	Author
v0.1	2017-06-05	Initial Version	P.Georgantas
v0.2	2017-06-09	Contributions on Software Components	K.Karpouzis
v0.3	2017-06-13	Contributions on User Models	C. Raftopoulou
v0.4	2017-06-15	Contributions on System Architecture	A.Symvonis
v0.5	2017-06-21	Various edits	P. Georgantas
v0.6	2017-06-27	Internal Review	C.Mihu
v0.7	2017-07-01	Internal Review	M.Karlsson
v1.0	2017-07-17	Final Version	P.Georgantas

Table of content

Table of figures

1. Executive Summary

This document presents the System Architecture of iRead, a project that aims to develop personalized learning technologies to support reading skills. The layout of this document is the following.

First, we introduce the Logical Architecture of iRead. The main software components that constitute the system are identified along with their dependencies. We continue by describing in detail each component and its functionalities.

Afterwards, we discuss the Physical Architecture of iRead. We present an overview of the designed architecture and proceed to a more detailed view of each subsystem with an emphasis on data storage. The main choices are presented and the way they contribute to iRead's open and scalable infrastructure is explained.

Finally, for each of the discussed software components of the iRead Infrastructure, we identify the APIs that are exposed to the client applications. For each service API we examine in detail how it is called, its parameters and the results returned.

2. Logical Architecture

In this section, we present an overview of the iRead logical architecture and describe in detail the main components.

2.1. Architecture Overview

The proposed architecture is designed using a Client Server model and based on the Service Oriented Architecture. Clients use the network to consume services provided by Server components using specific APIs (Application Programmable Interfaces).

The iRead System main architecture is pictured in the following figure.

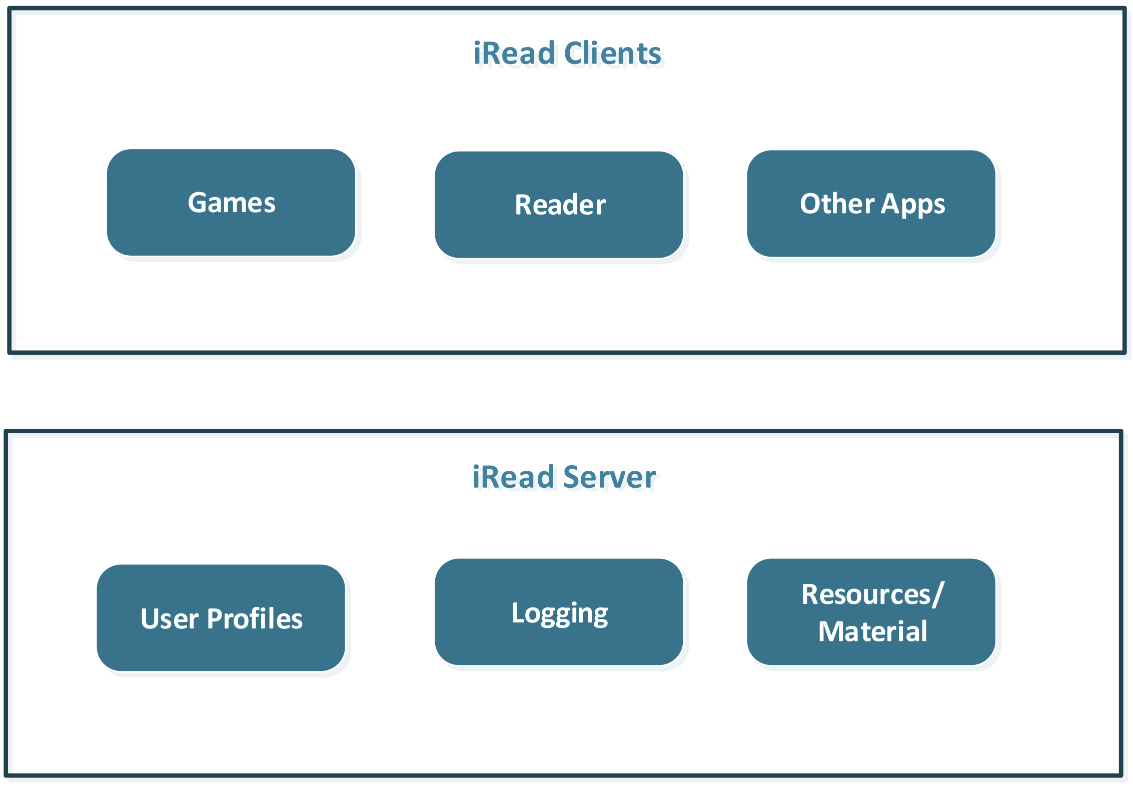

Figure 1 Architecture Overview

On the server side, we have the iRead Infrastructure which provides to the clients the necessary services to communicate with the iRead System. The top logical components of this infrastructure are:

- User Profiles: Provide information about a user, access to one or more profiles for this user stored in iRead and his progress on them.
- Logger: A facility for logging user activity and progress.
- Resources: Provides content for the applications using iRead. This content can be words, documents or multimedia items that are related to specific problems of a profile or have certain properties.

Also, on the server side there are the server components of the iRead System's clients. These are not part of the iRead Infrastructure since each one only serves specific clients' needs.

On the client side, there are the iRead Applications used by the users. The applications run on mobile devices or external systems and use the iRead Infrastructure's services to communicate with the platform. These applications are:

- Games supporting reading skills.
- The Reader application
- 3rd party Applications that take leverage of iRead Infrastructure to provide personalized learning for the users.

2.2. Components

In a more detailed view of the iRead Infrastructure, we have identified the software components that appear in the following figure and are described below.

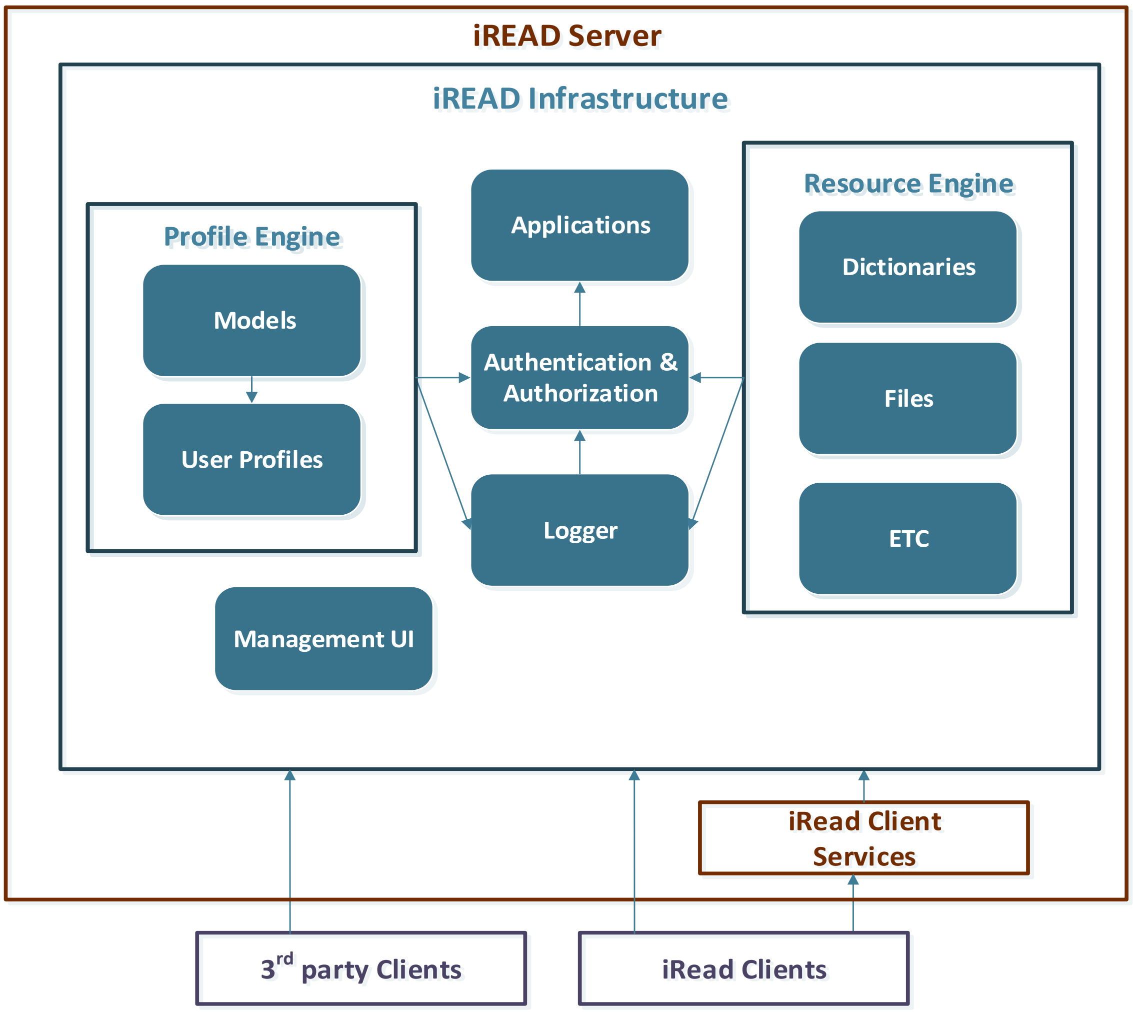

Figure 2 Software Components

2.2.1. Profile Engine

The Profile Engine is the core part of the iRead Infrastructure. It implements the representation of domain knowledge as a model, the management of user profiles that are based on these models and the users' progress on these profiles.

Models

The knowledge for a particular domain is represented in iRead by a Domain Model. Such domains could be learning to read for children with dyslexia, or learning English as a foreign language. Domain Models are implemented as Directed Acyclic Graphs (DAG). The models will be imported in iRead Infrastructure as graphs that comply with these requirements, that is not have cyclic dependencies and have directed vertexes.

Each node in the graph represents a specific problem that is to be addressed and contains all the required information about it. Edges between nodes represent the requirements for transitioning from one problem to another. For example, an edge could specify that in order to be able to progress from the parent node/problem to the child node/problem, the user must have achieved on the parent node a competence level greater than a certain value. Each edge also has a weight value which represents its significance for progressing to the next problem.

The nodes also have an enablement function which determines whether a node is reachable based on the satisfaction of the requirements of the edges that point at him, in combination with the weights of these edges.

User Profiles

A user profile is an instance of a specific domain model for that user. Each user can have many profiles, one for dyslexia, one for English as a foreign language, etc. The user profile is also a DAG and gets initiated as a copy of the domain model with domain specific initialization values.

Starting with a new profile, a user can only work on a subset of the problems contained in the profile, determined by the initialization values dictated by the domain model. When the user uses the iRead applications to work on a specific problem, the node in the relevant profile that represents this problem gets updated with his current competence level by the profile engine. For each node/problem in the user's profile, its accessibility for the user is determined by

a) Which edges that point to this node have their requirements satisfied, which in turn depends on the user's competence level on parent nodes

b) The node's enablement function as determined by the domain model.

Therefore, though the user profile graph is a copy of the full domain model graph, the user's progress on this profile is at any point depicted by which problems are accessible to him.

In the following figures, a user's progress in his graph is visualized. In the first figure, an instance of a user's profile is shown. The user can play with all the features of his model, except Feature 8. He has not had any progress on the features 4,5,7.

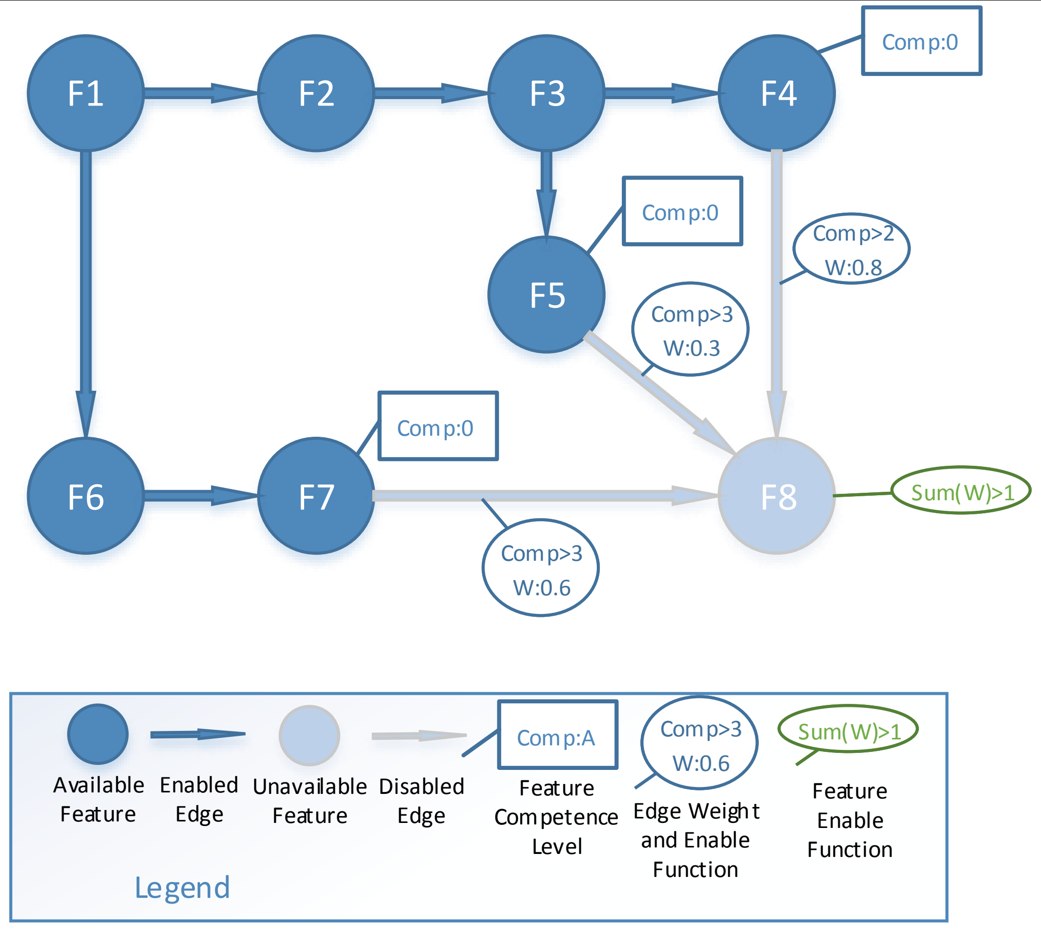

Figure 3 User Progress 1

In the following figure, the user plays a game concerning feature 5 and gained competence level 4 on this feature. This enables the edge from Feature 5 to Feature 8. However, Feature 8 is not yet active and thus not available to the user. Its enablement function, which requires the sum of the weights of the active edges to be more than 1 is not satisfied.

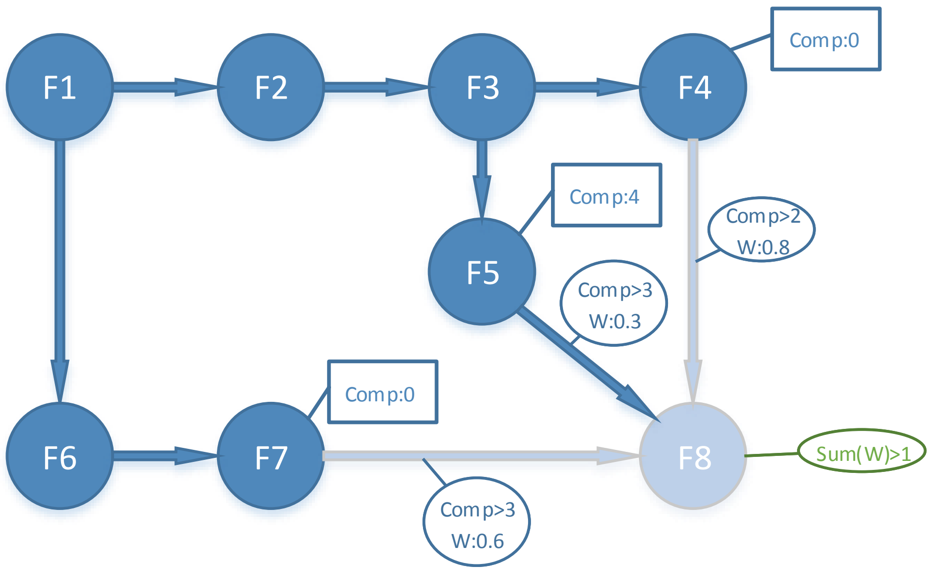

Figure 4 User Progress 2

In the last figure, the user has gained competence level 3 on Feature 4. This enables the edge between Features 4 and 8 and in return enables Feature 8.

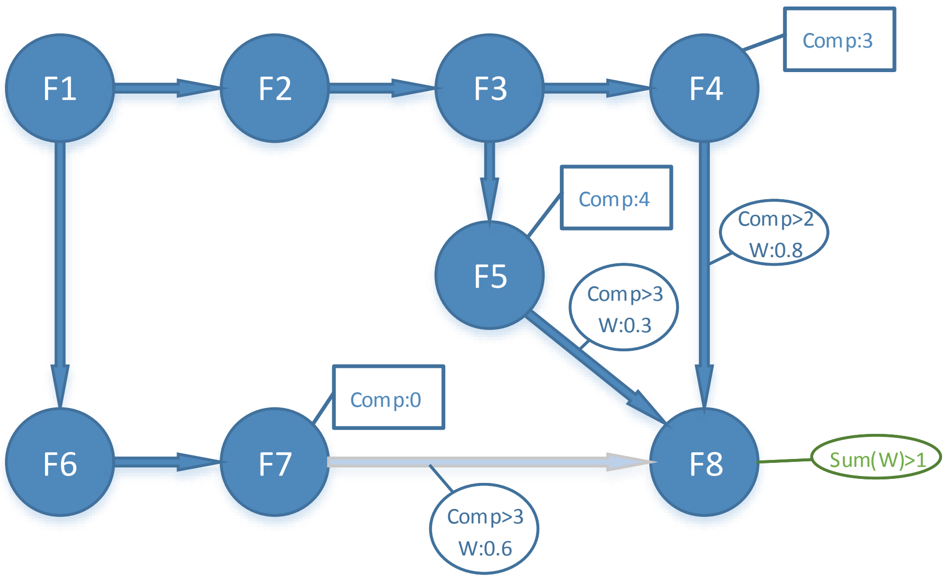

Figure 5 User Progress 3

2.2.2. Logger

In iRead Infrastructure the logging facility has an integral role. All user activity is logged and is treated as a valuable source of information. This information can be used to

- Provide the user with varying content
- Draw conclusions about the user's progress on problems
- Monitor and provide analytics about the users' progress.

The Logger is also used by the applications to log application specific information (errors etc.) and the iRead Infrastructure itself for its logging needs.

2.2.3. Resource Engine

This component acts as central repository which provides content to the applications. This content can be

- Words: Various dictionaries can be served using this component.
- Documents: Preprocessed annotated text.
- Multimedia Files.

The content inserted in the Resource Engine has properties describing its characteristics, and can be associated with specific problems in existing Domain Models. In the iRead architecture the Resource Engine component is a separate component which when queried can provide content

a) Independently from the rest iRead infrastructure, by making requests querying directly on the resources properties

b) Using the Profile Engine's Domain Models information, serving resources related to specific problems.

c) By taking advantage of the Logger, to provide content suggestions depending on the user's previously seen content.

2.2.4. Authentication/Authorization

Each user of the iRead infrastructure is assigned a username password combination which they can use to authenticate within the iRead system. This component is responsible for verifying the user's credentials and control what level of access is granted to the user.

In iRead security and user privacy are very important goals. The users' passwords are never stored in their plaintext version, instead a secure cryptographic hash of the password is kept. The hash is computed using the Argon2 [1] algorithm, which is a very secure key derivation function that won the Password Hashing Competition in 2015 [2].

In order to satisfy the project's goal for an open infrastructure consisting of interoperable components, while at the same time ensure user privacy and system security, for user authentication the Authentication and Authorization component takes leverage of OAuth 2.0 [3], the popular open standard for access delegation. OAuth 2.0 enables users to securely delegate access to their profiles to iRead Clients without having to share their credentials. It works using the HTTP protocol which makes it available to virtually any client, and uses Transport Layer Security [4] (TLS) to provide end-to-end encryption between clients and the server. OAuth 2.0 is a specification that covers a wide range of authentication and authorization options, not a strict protocol. In iRead we will implement only the flows of OAuth 2.0 useful in our context.

Authorization is another important part in iRead Infrastructure especially given the fact that there are superusers (teachers) that can have special permissions on other user profiles or groups of user profiles (classes). Authorization is handled in a central manner by the Authentication and Authorization Component which controls whether every incoming user request on any module is allowed or not. This control is implemented using the Role Based Access Control concept. Users belong to roles, and roles are granted privileges on other users or resources. For example a teacher could have the update_contact_info privilege on a group of users that are his students.

2.2.5. Applications

The Applications component is management component responsible for the coordination of applications who use the iRead Infrastructure. It stores and makes available information about these applications like their IDs in the iRead system or the secrets used for OAuth 2.0.

Each application is defined as a collection of games, where each game is associated to one or more items of a domain model. Each application is assigned a unique id within the iRead Infrastructure. In order to allow reuse of the same games between applications, each game is also assigned a unique id which can be shared between applications. When reporting information about user activity the applications must use both the application Id and the game id concerning this activity.

2.2.6. Management Component

The iRead Infrastructure also includes a Management Component which exposes some services and offers a user Interface for administrative purposes. An authorized user can use this UI to perform the following actions:

1. Model Management: Import new Domain Models into the system.
2. User Management: Create new users or modify users that are under his control.
3. Profile Management: Enable profiles for existing users that he is authorized to.
4. Analytics: View analytics about users and their progress.
5. Application Management: Define new applications or games or modify existing ones.
6. Resource Management: Import new resources or modify existing ones.

2.3. Use cases

The use cases of the iRead system are described in deliverable 3.3. A few use cases of the iRead system clients mentioned in section 2.1 and how the proposed architecture supports them are described below.

2.3.1. Student plays with iRead Games

1. Student logs in the literacy games application. The game application uses the Authentication/Authorization module to login the user and create an authorization token for the application.
2. Game “gains access” to user-model and usage-data by requesting them from the Profile Engine using the authorization token.
3. Game requests from the Profile Engine the next model feature to be addressed.
4. Game requests from the Resource Engine the literacy material related
5. The student starts playing the activity and the application records his actions
6. When student finishes with the activity, the game uses the logger API to log his actions and usage data.

2.3.2. The Reader application

1. Student logs in the Reader-app. The Reader application uses the Authentication/Authorization module to login the user and create an authorization token.
2. Student selects a document to read and the iReader app loads it using the Resource Engine API.
3. The Reader-app uses the Profile Engine API to query for user preferences and displays the book/document in a user-specific manner.
4. The student “reads” the document and “interacts” with it.
5. On exit, the Reader-app uses the Logger API to store data related to the user’s reading.

2.3.3. 3rd party Application. An eBook store with personalized search functionality.

1. The creator of the eBook store app uses the iRead Infrastructure's Applications Component to register his app and get an OAuth 2.02 secret.

2. User accesses the eBook store app.

3. The book store app uses OAuth 2.02 and the Authorization Module to request access to user preferences or profile. The user logs in the iRead System without the 3rd party app seeing his credentials and authorizes the eBook store app to access his profile.

4. The user issues a search query to the bookstore collection

5. The book store app uses the Profile Engine to retrieve helpful information from the user profile

6. The query results are ranked and presented in "easiest-to-read first" order based on the user's user-model (the ranking used the "difficulty score" of the books)

7. The user selects an eBook from the query results and details on the "difficulty/easiness of reading" are displayed (in some interactive graphical form)

3. Physical Architecture

In this section, the physical architecture of iRead Infrastructure is presented. The architecture is described in the context of a *cloud-based* installation using Virtual Machines. First an overview of the architecture is provided. Next the physical architecture and especially data storage design decision are explored in more detail.

3.1. Overview

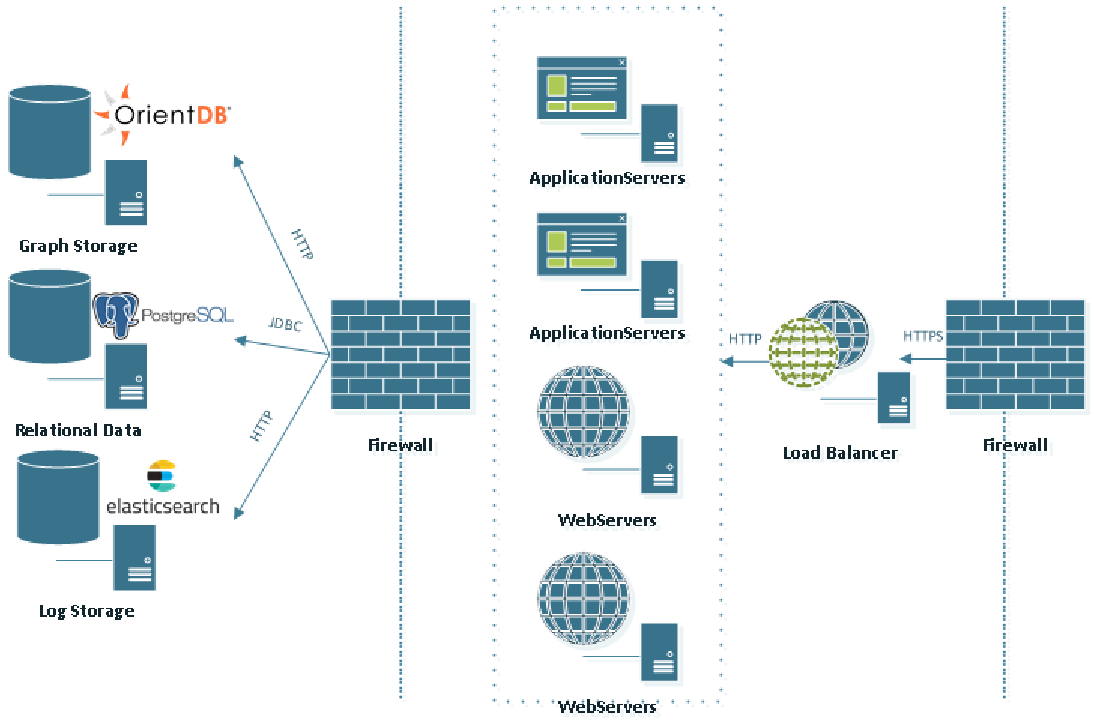

Figure 6 Physical Architecture

The iRead Infrastructure physical architecture overview is illustrated in Figure 3.

A set of web and application servers is placed in the frontend of the system. These serve the iRead Infrastructure applications. More specifically they expose over HTTPS the iRead Infrastructure services to the clients that consume them and serve the Management Web UI to the users.

A set of data storage servers are sitting in the back of the architecture and used by the application servers for storing user and application data. The iRead Infrastructure uses three types of data stores:

1. Graph Database for storing Domain Models and User Profiles
2. Relational Database Management System for storing frequently changing data
3. Log Storage and Search Engine for storing the iRead activity and application logs

3.2. Model and Profile Storage

Domain Models in iRead are the representation of the knowledge for a particular domain and are implemented as Directed Acyclic Graphs. A User Profile for a specific domain is an instance of the Domain Model Graph for this user.

Graphs are generally very hard to implement in the classic Relational Database Model and these implementations frequently suffer in performance. In iRead we decided to use a specialized Graph Database for storing Domain Models and User Profiles. A Graph Database is a database that uses graph structures for semantic queries with nodes, edges and properties to represent and store data. In graph databases the relationships between data items are a key concept in contradiction with relational databases where links between the data are stored in the data themselves and use the join concept to find related data.

As a graph database iRead uses OrientDB [5] which is an open source NoSQL database management system written in Java that supports graph models among others. OrientDB satisfies the requirements for the iRead open and scalable Infrastructure since it

- Is fully transactional and supports ACID [6] transactions guaranteeing that all database transactions are processed reliably
- Offers a native management of graphs
- Is distributed and supports multi-master replication
- It is free and open-source, released under the Open Source Apache 2 [7] license

3.3. Relational Data storage

A graph database is a perfect fit for storing graphs such as the Domain Models, however the iRead Infrastructure uses many other types of data that are best represented and stored using the Relational Database Model. Such data are but not limited to user information, role membership, application metadata. PostgreSQL [8] is used in iRead for storing these data. PostgreSQL is one of the most popular Relational Database Management Systems that delivers high performance and scalability. PostgreSQL offers the following features that support iRead's open and scalable infrastructure:

- Is fully transactional and supports ACID transactions guaranteeing that all database transactions are processed reliably
- Supports a wide variety of advanced data storage and query features.
- Supports replication and sharding which guarantees horizontal scalability and high availability
- It is free and open-source, released under the terms of the PostgreSQL License [9].

It should be noted that OrientDB offers a document database except from being a graph database, therefore it could be used to store data like use profiles instead of bringing a secondary Data Storage system in iRead Infrastructure. However, we chose using a Relational Database Management System in order to take advantage of the existing expertise and available software modules compared to implementing everything on the less widely supported OrientDB.

3.4. Log Storage

Logging plays a very important role in the iRead Infrastructure. User actions are logged from various clients and processed in order to provide better experience for the users and also derive conclusions about the users' progress. At the same time, the logging facility is used by client applications for their needs. The diverse nature of the logged content in iRead promotes the use of storage engine that offers good support for semi-structured content while offering high performing search functionality. Elasticsearch [10], a search engine base on Apache Lucene [11], a high-performance, full-featured text search engine library, was chosen for this purpose. Elasticsearch supports schema-free JSON documents and handles especially well not frequently changing data such as logs. Important features of Elasticsearch within the iRead Infrastructure are:

- It has very strong analytical search capabilities
- It is horizontally scalable since its indices can be divided into shards and distributed across several nodes with rebalancing and routing done automatically
- It inherently supports replication offering high availability and increased performance
- It supports multitenancy, which is important for organizations that want to use iRead system and create their user base.
- It is free and open-source, released under the Open Source Apache license

4. Offline Mode

The iRead System's Client applications will provide an offline mode where access to the internet will not be available and some reduced functionality will be provided to the user. The proposed Architecture covers the need for such offline functionality. All operations are designed so that they are

- initiated by the clients at their discretion,
- atomic and not require a specific order or have dependencies
- respond with blocks of information the client can cache for his purposes.

When a student or a teacher initiates offline mode in an application, the application is responsible for requesting all the necessary user-data and resources in order to be able to provide the desired offline functionality. When the application comes back online it will send to the Logger component all the user activity logged in offline mode.

In the context of iRead Infrastructure's architecture offline mode may require special attention, depending on the client implementation, when it comes to how user authentication is performed when offline mode is used. The following options exist:

1. In Offline mode, no user authentication exists. The person initiating offline mode will be the only one with access to the device until connectivity is restored
2. Offline mode is initiated for specific users and the client application maintains a separate userbase and distinct credentials for these users. The client application is then responsible to maintain a match between its own users with the iRead Infrastructure users.
3. Offline mode is initiated for specific users. For these users, the client application retrieves from the server their password hashes, therefore is in place to perform offline authentication of each user. In this case special care must be taken to use a very strong cryptographic algorithm for storing the passwords, like the Argon2 algorithm discussed in section 2.2.4.

5. API

In this section, we describe the services offered to the iRead Infrastructure clients. These services expose over HTTPS a simple RESTful API which facilitates their consumption by any type of client, regardless of the client technology or device. The exact final specifications of these APIs will be included in Deliverable 9.2 or an updated version of this document.

5.1. Authentication - Authorization

The iRead Infrastructure uses the OAuth 2.0 framework to accomplish authentication and authorization. According to OAuth 2.0 each client application gets registered and acquires client ID. This for iRead is a manual process since 3rd party applications need to be approved before using the Infrastructure. A client secret is also provided to web applications in contrast to mobile or desktop apps where such a secret would be redistributed and therefore not considered to add any meaningful security. The client application can use these credentials along with the user's permission to acquire an access token that will allow it to access user resources as permitted by the Authorization Module. Figure 4 shows a basic authentication flow in OAuth2.0 environments.

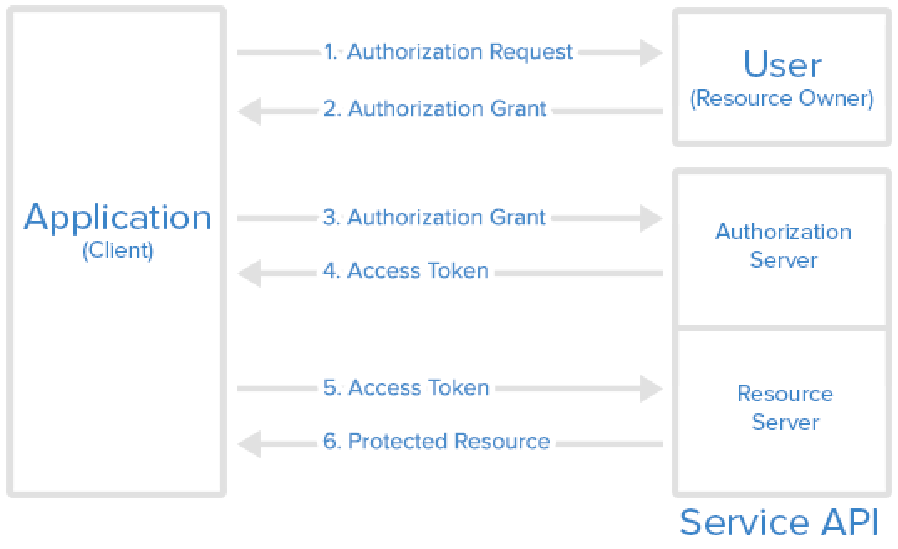

Figure 7. OAuth 2.0 Authentication Flow

The following APIs are available for the Authentication and Authorization Component.

5.1.1. Get Access token

It is used by the client to get an access token by prompting the user to enter his credentials.

Url: /oath/access_token

Method: POST

Parameters:

- Client_id: the registered id of the client application

- Client_secret: the client’s secret if one is required
- Grant_type: “client_credentials”

Return Values:

- Access_token: The access token granted
- Expires_in: expiration of token
- Refresh_token: A possible refresh token to be used for refreshing the access token before it expires.

5.1.2. Get Authorization code

For clients that do not rely on a user entering his credentials an access_token can be granted with the following API.

Url: /oath/access_token

Method: POST

Parameters:

- Client_id: the registered id of the client application
- Client_secret: the client’s secret if one is required
- Grant_type: “authorization_code”

Return Values:

- Access_token: The access token granted
- Expires_in: expiration of token
- Refresh_token: A possible refresh token to be used for refreshing the access token before it expires.

5.1.3. Refresh token

It can be used by the client to refresh its access token using the refresh token granted to it.

Url: /oath/access_token

Method: POST

Parameters:

- Client_id: the registered id of the client application
- Client_secret: the client’s secret if one is required
- Refresh_token: The token to be refreshed
- Grant_type: “refresh_token”

Return Values:

- Access_token: The access token granted

- Expires_in: expiration of token
- Refresh_token: A possible refresh token to be used for refreshing an access token.

5.2. Profile Engine

The Profile Engine Component is responsible for maintain user profiles and exposes the following APIs:

5.2.1. Get User Info

It is used by the client to get information (Name, email) about the user.

Url: /user/info

Method: GET

Parameters:

- Access_token: the access token granted to the client.
- Uid:the user id
- Info_id: Optional, an array of the particular information requested (name, email etc.)

Return Values:

- A JSON document containing the user information

5.2.2. Create new user profile

It can be used to create a new profile for an existing user based on a domain model.

Url: /user/profile

Method: PUT

Parameters:

- Access_token: the access token granted to the client.
- Uid:the user id
- Domain_Model_ID: The id of the domain model that this user profile is based on.

Return Values:

- The Profile_Id for the created user profile.

5.2.3. Delete user profile

It can be used to delete an existing user profile.

Url: /user/profile

Method: DELETE

Parameters:

- Access_token: the access token granted to the client.
- Uid:the user id
- Profile_ID: The id of the domain model that this user profile is based on.

Return Values:

- OK or error.

5.2.4. Get User Profile Preferences

It is used by the client to get stored user preferences for a certain profile.

Url: /user/profile/preferences

Method: GET

Parameters:

- Access_token: the access token granted to the client.
- Uid:the user id
- Profile_Id: the profile's id.
- Preference_id: Optional, an array of the particular preference requested

Return Values:

- A JSON document containing the user preferences requested.

5.2.5. Set User Profile Preferences

It is used by the client to set stored user preferences for a certain profile.

Url: /user/profile/preferences

Method: PUT

Parameters:

- Access_token: the access token granted to the client.
- Uid:the user id
- Profile_Id: the profile's id.
- Preference_id: The preference to be set
- Preference_value: The preference value.

Return Values:

- OK or error.

5.2.6. Get user profile item

It can be used to get information about a feature in user's profile.

Url: /user/profile/item

Method: GET

Parameters:

- Access_token: the access token granted to the client.
- Uid:the user id
- Profile_ID: The profile id.
- Item_ID: Optional, an array of the profile items requested.

Return Values:

- A JSON document containing the profile items requested.

5.2.7. Update user profile item

It can be used to get information about an item in user's profile.

Url: /user/profile/item

Method: POST

Parameters:

- Access_token: the access token granted to the client.
- Uid:the user id
- Profile_ID: The profile id.
- Item_ID: The ID of the updated item.
- Item values: An array of attribute value pairs for the profile item.

Return Values:

- OK or error.

5.2.8. Get next profile item

It can be used to get the next available feature(s) in a user's profile.

Url: /user/profile/next

Method: GET

Parameters:

- Access_token: the access token granted to the client.
- Uid:the user id
- Profile_ID: The profile id.

Return Values:

- A JSON document containing the next available profile items.

5.3. Logger

The logger component exposes the following APIs:

5.3.1. Log Action

It can be used to log user actions.

Url: /log/action

Method: POST

Parameters:

- Access_token: the access token granted to the client.
- Uid:the user id
- Timestamp: The time this action took place.
- Data: A JSON document containing the data to be added to the log.

Return Values:

- OK or error.

5.3.2. Log Application Messages

It can be used to log application messages.

Url: /log/application

Method: POST

Parameters:

- Access_token: the access token granted to the client.
- Uid:the user id
- Timestamp: The time this action took place in ISO 8601:2004 format.
- Data: A JSON document containing the data to be added to the log.

Return Values:

- OK or error.

5.3.3. Get last action

It can be used get the last user actions.

Url: /log/lastaction

Method: GET

Parameters:

- Access_token: the access token granted to the client.

- Uid:the user id
- Data: A JSON document containing parameters to filter which actions are to be returned.

Return Values:

- A JSON document containing the last actions.

5.3.4. Get last Material

It can be used get the last user material.

Url: /log/lastMaterial

Method: GET

Parameters:

- Access_token: the access token granted to the client.
- Uid:the user id
- Data: A JSON document containing parameters to filter what kind of material are to be returned.

Return Values:

- A JSON document containing the last material the user accessed.

5.4. Resource Engine

The resource engine serves content that can be used in iRead client Applications and exposes the following APIs:

5.4.1. Dictionary

It can be used return words and information about them from a dictionary.

Url: /resources/dictionary

Method: GET

Parameters:

- Access_token: the access token granted to the client.
- Dictionary_ID: The dictionary to be accessed.
- Word: An array of words to be returned.

Return Values:

- A JSON document containing the requested words and the available information about them.

5.4.2. Resources for profile item

It can be used return available resources for a specific profile item.

Url: /resources/profileitem

Method: GET

Parameters:

- Access_token: the access token granted to the client.
- Domain_Model_id: The domain model of the profile
- Profile_Item_Id: The id of the profile item.
- ContentType: The desired content types.

Return Values:

- A JSON document containing a list of available resources and information about them.

5.4.3. Get resource

It can be used to download the actual resource content.

Url: /resources/download

Method: GET

Parameters:

- Access_token: the access token granted to the client.
- Resource_Id: The Id of the requested resource.
- ContentType: Optional, possible desired content type

Return Values:

- The resource's content

5.5. User Management

The Management Component of the iRead Infrastructure offers a UI for managing users. It also exposes the following APIs that can be used from any other system to create users in the iRead Infrastructure, provided that an authorized superuser account is used:

5.5.1. Create User

It can be used to create new users in an organization by an external application.

Url: /manage/user

Method: PUT

Parameters:

- Access_token: the access token granted to the client.

- Uid:the user id
- Info: A JSON document with attributes and values for this specific user.

Return Values:

- A JSON document containing the user id of the created user.

5.5.2. Delete User

It can be used to delete users from the iRead Infrastructure.

Url: /manage/user

Method: DELETE

Parameters:

- Access_token: the access token granted to the client.
- Uid:the user id

Return Values:

- OK or error.

5.5.3. Create User group

It can be used to create a new user group (e.g. class).

Url: /manage /group

Method: PUT

Parameters:

- Access_token: the access token granted to the client.
- Info: A JSON document with attributes and values for this specific group.

Return Values:

- A JSON document containing the user id of the created user.

5.5.4. Delete User group

It can be used to delete user group.

Url: /manage/group

Method: DELETE

Parameters:

- Access_token: the access token granted to the client.
- Group_id:the group id

Return Values:

- OK or error.

5.5.5. Add Users to group

It can be used to add existing users to existing groups.

Url: /manage/group

Method: POST

Parameters:

- Access_token: the access token granted to the client.
- Group_id:the group id
- User_id: An array of user ids to add to the group

Return Values:

- OK or error.

References

[1] "https://en.wikipedia.org/wiki/Argon2".
[2] "https://password-hashing.net/".
[3] "https://oauth.net/2/".
[4] "https://en.wikipedia.org/wiki/Transport_Layer_Security".
[5] "http://orientdb.com/orientdb/".
[6] "https://en.wikipedia.org/wiki/ACID".
[7] "https://www.apache.org/licenses/LICENSE-2.0".
[8] "https://www.postgresql.org/".
[9] "https://www.postgresql.org/about/licence/".
[10] "https://www.elastic.co/products/elasticsearch".
[11] "https://lucene.apache.org/core/".