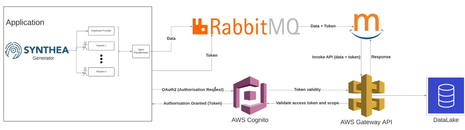
OOSection.md

OO Design & UML

High level architecture diagram



1. **INTRODUCTION ARCHITECTURE**

We propose the design of a client application which will:

1. Generate Data in csv standard
2. Separate data (Pseudomys data)
3. Integrate and Centralise Data
4. Use AMQP server to filter data
5. Create a safe, fast, and efficient connection with web services
6. **GENERATE DATA**

SyntheaTM

We use open source SyntheaTM Patient Generator to generate as-live data to simulate the regional healthcare landscape. Specifically, the simulators will generate data from a specific region:

- Simulate more than one healthcare provider category (e.g. Acute hospital, primary care, 111):
  - Patients
  - Conditions

Synthetic patients can be simulated with models of disease progression and corresponding standards of care to produce risk-free realistic synthetic health care records at scale.

The framework for the synthetic data generation process utilized by Synthea is based on the use of PARSER, the Publicly Available Data Approach to the Realistic Synthetic EHR.35 The PADARSER framework, unlike EMERGE25 and medGAN,27assumes that access to the real EHR is impossible or undesirable, relying instead on publicly available datasets to populate the synthetic EHR. Figure 1 presents the PADARSER framework.

HL7 FHIR Fast Healthcare Interoperability Resources (FHIR, pronounced "fire") is a standard describing data formats and elements (known as "resources") and an application programming interface(API) for exchanging electronic health records(EHR). The standard was created by the Health Level Seven International (HL7) health-care standards organization.

1. **DATA INTEGRATION AND CENTRALISATION**

We propose to use Lyniate Rhapsody Data Centralisation and Integration Engine.

MIRTH NextGen Connect is a cross-platform interface engine used in the healthcare industry that enables the management of information using bi-directional sending of many types of messages. The primary use of this interface engine is in healthcare.

Benefits of using Mirth are:

• It is built for Healthcare

• It has purpose-built solution for csv and FHIR

• It supports Data Acquisition (large amounts of data from multiple sources) - AMQP server

1. **Data Transfer Protocols**

Message broker technology (RabbitMQ) is an intermediary computerprogram module that translates a message from the formal messaging protocol of the sender to the formal messaging protocol of the receiver. Message brokers are elements in telecommunication or computer networks where software applications communicate by exchanging formally defined messages.

HTTPS is used for secure communication over a computer network, and is widely used on the Internet. In HTTPS, the communication protocol is encrypted using Transport Layer Security (TLS) or, formerly, Secure Sockets Layer (SSL).

1. **Data Ingestion**

The system will authenticate and create a RESTful endpoint for HL7 FHIR messages.

HL7 FHIR endpoint describes the technical details of a location that can be connected to for the delivery/retrieval of information. Sufficient information is required to ensure that a connection can be made securely, and appropriate data transmitted as defined by the endpoint owner.

RESTful API is an architectural style for an application program interface (API) that uses HTTP requests to access and use data. That data can be used by using the CRUD approach: create, read, update, and delete.

RabbitMQ is a messaging system that uses AMQP 0.9.1 as the basis for a set of standards controlling the entire message passing process.

Benefits of RabbitMQ:

1. Delivery and order guarantee: The messages have been sent to a consumer in the same order in which they were created.
2. Redundancy: The queues persist the messages until they are processed completely.
3. Decoupling: Any third party system can consume the messages and interact with them, so you want the messages to be processed by someone who is not the actor who created the message, without any problems. This generates us a benefit, which is that it can be reused for many projects.
4. Scalability: we can have an application server dedicated to the processes and the other servers for browsing the web.

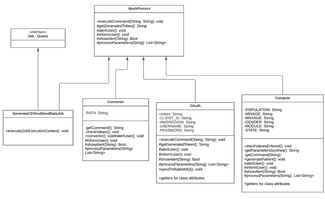
OAuth 2.0 is the industry-standard protocol for authorization. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices. This specification a nd its extensions are

Authentication will be secured by using **Amazon Cognito**. The system will use a secure Token to access the API Gateway to create a safe and recognized connection with the HealthCare Lake/database infrastructure. The API Gateway will run a RESTful API and a HL7 FHIR message for ingestion into data lake. Amazon Cognito will verify the token and continue with the data transfer.

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. Using API Gateway, RESTful APIs enables real-time two-way communication applications. API Gateway supports containerized and serverless workloads, as well as web applications.

Static UML modelling aspect

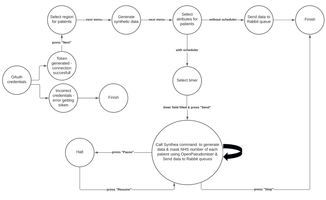
Class UML diagram



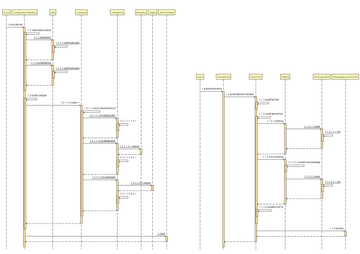
One of the most important aspects of the design of our project is related to the abstract class "BashProcess". We had several main classes in our project that were supposed to perform similar tasks such as executing a command depending on an attribute related to the region the patients are generated from, informing the user wheather the task has completed succesfully, processing parameters related to executing different jobs or alerting the user if conditions are not met.
As a consequence, in order to avoid duplicates in our code, we created this abstract class and have "OAuth", "Compute", "GenerateCSVAndSendDataJob" and "Convertor" extend it, allowing for the implementation of custom functionalities.

Dynamic UML modelling aspect

State Machine Diagram

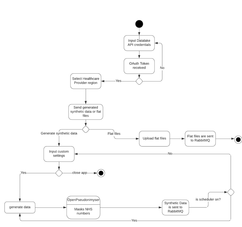


Here are some significant **Sequential Diagrams**:



The behavioral state machine is used to model the behavior of the menus, the system regarding the usage of the third party Synthea, of the OpenPseudomiser for masking data and the correlation of producing, masking and sending the data with the cron job.
The transitions represent how state changes. For our design the most important feature is related to the scheduler which rules how and when the Synthea should produce data and when the data is masked and sent to Rabbit queues.
When pressing "Pause", the system moves into a halting state. To escape this state, the user ought to press "Resume" and the application will keep on generating, masking and sending data until some other command is selected. Pressing "Stop" will cause the system to move into a finish state.
The sequence diagrams that accompany the State Machine Diagram picture how the classes and methods responsible for masking and sending data communicate over time and with the user interface.

Overview flow of the application - Activity UML diagram:



The purpose of this diagram is to have an overview of the **control flow**, showing the various paths that exist while the program is being executed. It was created to help us have a better understanding of the **sequential execution**, to provide a suggestive way to present how our project works, and **aid communication between developers and clients**. It changed many times during the developing every time a feature was added or modified, providing a useful vehicle to vizualize the system functionality without needing to read the code in detail.