qDocs: Citizen-Centered and Multi-Curator Document Automation Platform: The Curator Perspective

José António Trocado de Saldanha Sousa e Menezes
jose.menezes@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2019

Abstract

Document automation is a popular approach that supports the creation of electronic documents in a flexible and efficient way. These systems allow the definition and management of (document) templates, which are extended versions of common documents with particular elements called fields, merge fields, form objects, etc. In this paper we introduce and discuss qDocs, a citizen-centric and multi-curator document automation platform for managing dynamic electronic documents (e.g. id cards, forms, certificates) that are accessible to any citizen in a easy and secure way. qDocs provides a single point of access for citizens to create, use and manage their own documents. These documents are produced from templates curated by public or private organizations (named as curators) that also participate in this qDocs ecosystem. This paper discusses particularly how curators may define, design and configure their templates and then make them available to citizens, allowing access to their respective documents in a secure and flexible way.

Keywords: Document Automation; Document; Template; Citizen-Centered; Curator; Citizen.

1. Introduction

Document automation is a popular approach that supports the creation of electronic documents in a flexible and efficient way. These software systems allow the definition and management of (document) templates, which are extended versions of common documents with particular elements called fields, merge fields, form objects, etc. Document automation systems allow to create and assembly new documents from these previously defined templates by replacing form objects with concrete data collected directly from end-users or from external data sources. Nowadays there are some document automation tools and platforms (e.g., ActiveDocs, SmartDocuments, HotDocs or xPressDocx), but these are mainly focused on document assembly for just one organization purpose, thus not accessible at all to the end-users and, in particular, to citizens when they intend to interact with such organizations, in the scope of administrative or bureaucratic processes [18].

This paper introduces and discusses qDocs, a citizen-centered and multi-curator document automation platform for managing dynamic electronic documents that are accessible in a easy but secure way to any citizen through any device [17]. It has been developed and promoted by MDSS[1] with support and research developed by the Information and Decision Support System Group of INESC-ID[2].

The main purpose of qDocs is to provide citizens a single point of accessing and managing their own documents, like identification documents, forms, certificates, reports and questionnaires. These documents are provided by curators, which are private or public organizations that are connected to the qDocs platform. A curator creates its documents within the platform as templates that then become available for citizens belonging to it. qDocs is a collaborative platform that facilitates and promotes the relationship between citizens and curators through any process that involves the request, publication, access, delivery and sharing of electronic documents. The design of templates is defined in the scope of just one curator, but the orchestration of data services can come from different curators.

qDocs allows the search for documents based on several criteria such as kind of life events (e.g. birth, health or education), document type (e.g. certificates, forms or identification documents) or curator (e.g. Tax Services, University or Public Services).

Figure 1 presents the general architecture for

[1] https://mdss.pt/index.php, last accessed 30th July 2019
[2] https://www.inesc-id.pt/, last accessed 30th July 2019

the qDocs platform; which suggests the integration of four key applications: qDocs/Citizen, qDocs/Curator, qDocs/Admin, and qBox. The qDocs ecosystem involves the interaction of the qDocs platform itself, the citizens and the curators. Citizens interact through the qDocs/Citizen application. Curators create, define and configure templates through the qDocs/Curator app and then what delivers the data to the respective citizens is the qBox platform, which is the platform that integrates with all the curators' databases and applications. The qDocs/Admin application is used to manage and configure he general aspects of the qDocs platform.

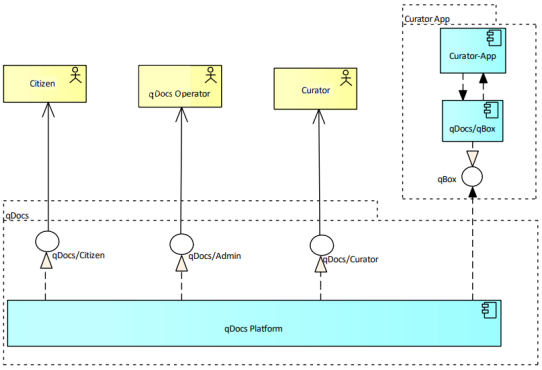

Figure 1: qDocs General Model (ArchiMate diagram)

Regarding data storage, data stored in one or more curators is dynamically merged in documents generated in real time. The fact that data storage does not leave its respective curator's datacenters, allows curators full control over their information. Document materialization only happens in the citizen's device, increasing privacy and information security. Only the citizen can manage and has full access over his own personal information.

For a citizen to have access to a document, the following happens in the qDocs ecosystem: first, the citizen requests a document to the qDocs platform; then the qDocs platform sends a "request acknowledged" message to qBox; after that, qBox sends a key to qDocs; then qDocs sends that key with the document template to the citizen's device; then the citizen's device sends the key directly to qBox; then qBox sends the respective data to the citizen and the document is generated in the citizen's device; Finally, qBox saves the data from the citizen so he/she can access the document anytime from that moment on.

qDocs is an existing document automation platform that is still in an early stage of development so it only provides basic features for curators and citizens, like template creation for curators and document access for citizens.

Since qDocs is supposed to be used by any kind of curator, it must support the creation of templates with different features and requirements, i.e. qDocs must support participant invitation for documents that require interaction from different citizens, and also features such as payment or strong authentication in order to access documents like certificates or forms. Also curator users are unable to configure settings for their own curator and every curator as access to every functionality within its curator, without any access restrictions.

Citizens in the qDocs platform must be able to interact with their documents according to what was defined by the curator that owns the document template, i.e. a citizen has to pay for a certificate before accessing it if the curator configures the certificate's template in that manner. Also, citizens must be able to invite participants to interact with their document or vice-versa. For example, the jury of a dissertation defense meeting must sign the respective minute of the meeting although the document does not belong to any of the jury's members.

Therefore, this paper aims to provide curators management and configuration capabilities within their curator, as well as flexibility regarding template configuration. The same goes for citizens, that must be able to interact with their documents according to the configuration made by the respective curators, including.

2. Background

This research has been influenced by the relevance and impact of document automation platforms in the efficiency of businesses, but also on their lack of being citizen-centered.

Figure 2 depicts the common processes and artifacts involved in the context of document automation platforms. First (P1), a template manager is responsible for the definition and design of document templates, which are stored in a persistent data storage for document templates. Second (P2), an operator or end-user uses the document templates together with information obtained from a data-source to assembly or produce specific documents, that can be stored in persistent data storage. Document automation (or document assembly) software tools intend to replace the manual filling of form-based documents with templates by allowing, for example, users answer software-driven interview questions. The information collected from end-users or external data sources can be used with such document templates to produce another set of concrete documents [22, 15, 6].

Document automation provides several benefits for organizations such as improved efficiency in document production (i.e. documents take less time to be produced), reduced human errors (i.e., end-users only have to provide the necessary information rather than producing a docu-

ment from scratch), costs (producing multiple documents takes less time, so costs are reduced) and turn-around times for clients (a document request can be fulfilled faster if a template already exists) [13, 14, 20].

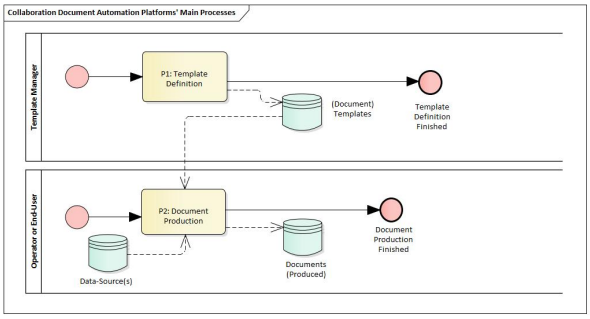

Figure 2: Document Automation Platforms' Main Processes (BPMN diagram)

Document automation platforms are becoming popular in several areas, such as law [16], public administration [2, 9], e-business [5] or even the aerospace industry [3]. Document automation is particularly popular in the business and legal services areas, since it is where a great number of repetitive documents (e.g. contracts, surveys, minutes) are produced [25, 27, 26]. For example, "Do-It-Yourself (DIY) law" is becoming a reality and is improved due to these document automation tools, since it became easier for self-representing litigants to produce documents they need [12]; for example, A2J Author[3] is a platform designed for this purpose [4]. Regarding e-government, document automation is also changing the paradigm from "one-size-fits-all" to "one user, one document" [16], enabling the generation of personalized documents in domains with high variability, having a template that can be reusable and slightly changed to different organizations [24].

Document automation tools keep evolving, for example, they are using machine learning techniques to validate and automatically create contracts [1], can provide visual information, such as charts and tables, based on the end-user or external data source information [23] and document templates are easier to change or upgrade [10].

To better understand the common features of these tools, an evaluation framework was defined, and involves the following dimensions: Document Template Definition, Document Generation, Content Management, Integration, Platform Availability, Digital Signature, Citizen-Centered, Other Features. These dimensions provide information about the architectural details of the platforms described. The citizen-centered dimension is the only one that has a subjective measurement, because there is no information about it but still is considered important since one objective of the qDocs platform is to be citizen-centered. The evaluated tools were the following: ActiveDocs Opus[4], Smart Documents[5], HotDocs[6], A2J Author[7], xPressDox[8], Templafy[9], Clio[10] and SmartDocs – Acculynx[11]. These tools share features that support processes like template definition, document generation and content management; they provide their services via Cloud, Server, APIs, or even Microsoft Office add-ins. Regarding template definition, some use MS-Word as template file format while others also use PDF. They support elements for template creation such as plain text, selection lists, text variables, repeating items and conditional blocks. Regarding document generation, some platforms may receive data from different data-sources as input. The generated documents are produced in multiple formats like PDF, HTML, XML, ODT or TIFF. Most of these platforms offer several integration possibilities and some have extra features such as digital signatures and customized workflow for documents.

3. Technology

qDocs is a Single Page Application (SPA) and uses two frameworks (ASP.NET Core and Angular) that integrate with one another. These frameworks use the following software programming languages: C#, Typescript (which is compiled to Javascript), HTML and CSS.

3.1. Single Page Application

SPAs are web applications or websites that are contained in a single web page. They are composed of individual components which can be updated independently, with the goal to be able to update parts of an interface without sending or receiving a full-page request, providing a more fluid user experience [11, 21, 19]. There is no refresh on page request, unlike in multi-page applications in which there may be a page reload on each request. All the necessary content is injected into the page through the use of Asynchronous Javascript and XML (AJAX) and HTML templates to render the content.

4. qDocs Functional Requirements

qDocs is defined by the integration of three applications: qDocs/Citizen aimed at any citizen looking for the benefits of the system; qDocs/Curator particularly oriented to the curators that make up

[3] https://www.a2jauthor.org/

[4] https://www.activedocs.com/
[5] https://smartdocuments.eu/en/
[6] https://www.hotdocs.com/
[7] https://www.a2jauthor.org/
[8] http://xpressdox.com/
[9] https://www.templafy.com/
[10] https://www.clio.com/eu/features/legal-documents/
[11] http://www.acculynx.com/

the qDocs ecosystem; and qDocs/Admin to manage and configure the available features for the qDocs/Citizen and qDocs/Curator applications.

Since qDocs is a platform that already had an initial development, the proposed solution was to further develop and extend the platform in the context of the curator. The development of this work is not restricted to the qDocs/Curator application, because to enhance the curator' user's experience, changes must be made to the whole qDocs ecosystem, therefore to the qDocs/Citizen and qDocs/Admin applications. The changes in the qDocs platform will be functional, keeping the architecture of the platform.

The following sub-sections present the functional requirements for each qDocs application, as well as the cross-cutting concerns of this work. Sections 4.1, 4.2 and 4.3 present the domain model of the respective application. Domain models have the same color scheme: **White** - if the class/use case already existed and was untouched throughout this work; **Yellow** - if the class/use case already existed but had technical modifications; **Green** - if the class/use case was created due to this work; **Grey** - if the class/use case does not exist in the platform but is scheduled for future work.

Finally, section 4.4 presents the considerations to take into account throughout this work.

4.1. qDocs/Admin

The qDocs/Admin application is used to manage and configure the available features for the qDocs/Citizen and qDocs/Curator applications — provides administration features for the qDocs ecosystem.

Figure 3 presents a UML class diagram of the domain model related to the qDocs/Admin application.

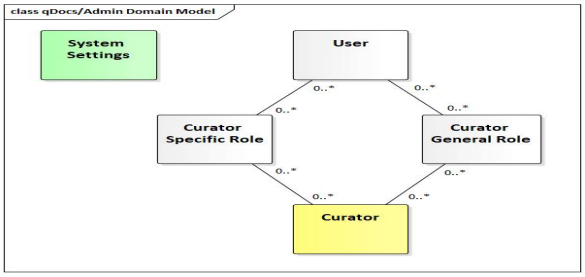

Figure 3: qDocs/Admin Domain Model (UML Class Diagram)

The **Users** class represents all users in the qDocs ecosystem. A **curator general role** represents a role with a certain degree of responsibility regarding a curator. A **curator specific role** represents a role that citizens may have regarding a certain curator. These roles are further explained in sub-section 4.2.1. Users may have multiple curator general roles and curator specific roles, and vice-versa.

The **Curators** class represents curators in the qDocs ecosystem. A curator must be associated with at least one curator specific role and one curator general role. These roles may be associated with multiple curators.

The **System Settings** class represents all the settings that are set for the qDocs platform, e.g. language and services available for the curators. This is a singleton class.

4.2. qDocs/Curator

The qDocs/Curator application is particularly oriented to the curators that make up the qDocs ecosystem — it allows the design of electronic documents (e.g. identification documents, certificates, reports, forms, questionnaires) based on the orchestration of services and data from different curators, managed according a distributed, dynamic and secure manner.

Figure 4 presents a UML class diagram of the domain model related to the qDocs/Curator application.

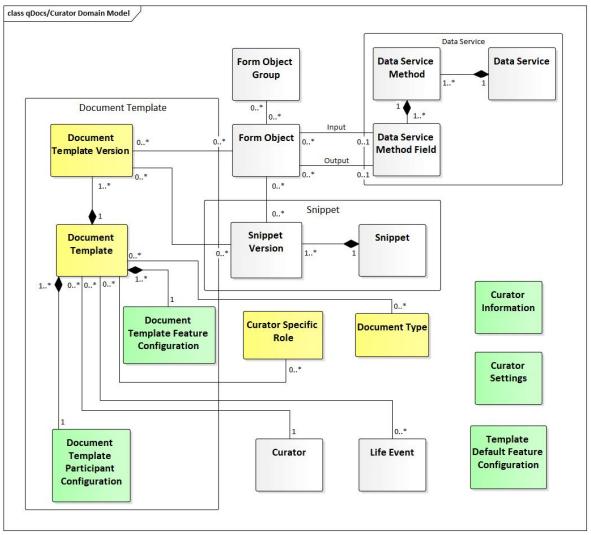

Figure 4: qDocs/Curator Domain Model (UML Class Diagram)

Document templates are represented by two classes: Document Template and Document Template Version. Document Template Version holds the information regarding a particular version such as date of creation and last update, and if the version is the one being used by the curator (if it is active or not). Document Template stores the basic information of a document template such as full and short names, and state and aggregates the first class. Document templates can have multiple versions and a document template version is associated with only one document template. Document Templates have an attribute called Template State, which is an enumeration and represents the current state of the document template. Document

templates can have multiple versions and a document template version is associated with only one document template.

Document Types are used to group document templates that have similarities, e.g. forms or certificates. A document type can have multiple document templates associated with it and vice-versa.

Life Events are used to group document templates that have similarities regarding certain life events, e.g. Birth or Marriage. A life event can have multiple document templates associated with it and vice-versa.

Curator represents the curator to which the document template is associated with. A curator can have multiple document templates associated with it and a document template can only be associated with one curator.

Specific Roles (also known as User groups) are used to group document templates to specific curator roles, e.g. an university diploma can only be accessed by students, which can be a specific curator role. A specific role can have multiple document templates associated with it and vice-versa.

A **data service** is provided by a curator through the qBox platform. It is composed by one or more data service methods, which organize the retrieved data in tables. Data service methods are composed by one or more data service method fields, which are the content of the table cells created by the data service method.

Form objects serve as a link between document template versions and data service method fields. They are associated with document template versions and receive or retrieve data from a data service method field, depending if the data service method field is an input or an output, respectively. The data received or retrieved by the data service method field is then used by the document template. Like document groups, form objects may be associated with **form object groups**, which group form objects that have similarities, e.g. input or output.

Snippets are blocks of text edited in the qDocs platform that may be used in document template edition. They can contain all the elements available in the qDocs template editor, such as form objects or even other snippets. Snippets are represented by two classes: Snippet and Snippet Version. Snippet Version holds the information regarding a particular version such as date of creation. Snippet stores the basic information of a snippet such as full and short names, and aggregates the first class. Snippets can have multiple versions and a snippet version is associated with only one snippet. Snippet versions may be associated with multiple templates and vice-versa.

Template feature configuration is comprised as the set of features that a document template version has enabled/disabled, e.g. Data Export or Strong Authentication. It has default values that are set by the administrator of the curator. A template feature configuration can be associated with many document template versions, but each document template version has only one template feature configuration associated to it.

Template participant configuration is comprised as the set of the participants a template version may have. Participants are entities that have a role towards a document, e.g. the president of a university must sign a student's diploma, meaning the president is a participant in the student's document in the qDocs context. Participants may be invited to access a document that was created by a citizen and perform the activities that are configured in the template participant configuration, Data Export or Sign.

Curator Information represents all the information regarding a curator, e.g. short name, full name and fiscal id. This is a singleton class.

Curator Settings represent all the settings that are set by the administrator of the curator, e.g. language and services available for that curator. This is a singleton class.

Template default feature configuration is comprised as the set of default features that a document template version has enabled/disabled, e.g. Data Export or Strong Authentication. This is a singleton class.

This work further develops the classes presented in this domain model, taking into account the different roles that each user may have.

4.2.1 Curator Roles

In the qDocs/Curator application, users can have general and specific roles. **Curator specific roles** are the ones that citizens have regarding a certain curator, e.g. in a university a specific role can be a student or a professor. Each specific role can request a determined set of documents to its curator. Each curator defines its own set of specific roles. On the other hand, **curator general roles** are associated with a certain degree of responsibility regarding a curator. Therefore, each curator general role has access to a specific set of features. These curator general roles are the following: Administrator; Data Manager; Templates Editor; Templates Manager; Auditor; Documents Manager.

The **Curator Administrator** is responsible for managing users at Curator-level and other Curator-level entities such as Document Groups, Curator Specific Roles and Form Object Groups. It is also responsible for managing users, namely associating users to the curator and managing a single

user's general and specific roles. Regarding user association, it can delete or add Users. Regarding management of other Curator-level entities, the Administrator can create, edit or delete such entities.

The **Curator Data Manager** is responsible for management Data Services and Form Objects. He can create, edit and delete Data Services. The creation of a data service comprises the creation of an association between the qDocs/Curator application and the curator's respective qBox platform. He can also create, edit and delete Form Objects, which are data objects that store information retrieved by data services and then are used in template authoring.

The **Curator Templates Editor** is responsible for management document templates and their versioning. He can create, edit, change and delete a document template, including its metadata, i.e. all information regarding a template except for its content, like current template version, short name, full name and state. He can create a new version of a document template after editing an "active" document template and can search and browse through document templates' previous versions. He can also manage the workflow of document templates and configure features for templates.

The **Curator Template Manager** is responsible for approve or reject document templates that are submitted for approval, therefore in the "Pending for Approval" state.

The **Curator Auditor** is responsible for analyzing Curator-level activity, i.e. can visualize dashboards and timeline graphics with the most relevant metrics at Curator level.

The **Curator Documents Manager** is responsible for the approval or rejection of document creation requests made by users (i.e., by citizens with general or specific roles). When a citizen requests a document creation, the Curator Documents Manager consults the request, and then it can approve or reject it. If the request is approved, a new document is created and the citizen (who requested the document) shall have access to it. If the request is rejected, the citizen shall be notified of the involved problem.

4.2.2 Template Authoring

The Curator Templates Editor can create and edit templates directly in the qDocs/Curator app. This editor has similarities to other text editors such as MS-Word, since it has features like text alignment, font size, font color and image insertion.

The difference in the qDocs/Curator editor is that it writes everything in HTML and it has particular features such as: Form Objects insertion, i.e. data fields that are going to be filled by citizen input (if the Form Object is of type input) or by curator data directly from his curator's qBox (if the Form Object is of type output); If/Else blocks, i.e. reusable text blocks that only appear if certain conditions are met; and Snippets, i.e. blocks of text pre-defined by the Curator Templates Editor that can include previous features and can be added to multiple templates.

4.2.3 Template Configuration

Besides template authoring in the qDocs/Curator editor, templates have metadata that can be viewed and configured by the Curator Templates Editor. This metadata is composed of the following elements: short name; full name; template version; creation date (i.e. the date of creation of the template's version); publication date (i.e. the date of activation of the template's version when the template's version becomes active); available feaures; document groups (i.e. life events, general document types and curator specific document types); user groups (i.e. curator specific roles); scope (i.e. if the template's version scope is defined as internal, the template can only be accessible to curator users; if defined as external the template is accessible by any citizen); and state (related to the template's workflow).

A template can have multiple versions: a new template version is created whenever the content of that template is edited and saved. Curator Templates Editors can always have access to the template's previous versions.

The features that a template can have are the following, which are enabled or disabled by the Templates Editor: Document Share — the document can be shared with other citizens via hyperlink or access key; Document Data Export — the document's data can be exported to a PDF, CVS or JSON file format; Request Strong Authentication — for a citizen to perform a specific action to a document (like create, access, share), he has to go through an authentication process; Request Payment — for a citizen to perform a specific action to a document (like create, access, share), he may have to pay a specific value defined by the curator.

If a template has one or more user groups (i.e. curator specific roles) assigned to it, then only citizens that belong to those groups may create or access the respective document. If the document has no user groups assigned to it, then by default any citizen can create or access that document.

Document groups are used to categorize documents, which are further divided into the following classifiers: life events; document types; and curator specific document groups. These classifiers

allow citizens to better filter and search for documents.

4.2.4 Template Workflow

A workflow is comprised as a set of states that are associated with a document template. There are several states for document templates. They are represented in Figure 5 and their sequence happens as follows: **Created** — after creating a document template; **Pending for Approval** — after the document template is designed it can be submitted for approval or if it was rejected it can be submitted for approval again; **Rejected** — if the responsible for approving the document template rejects it; **Active** — if the responsible for approving the document template approves it or if the responsible for the document template activates it after it is inactive or deprecated; **Deprecated** — when a new document template version is activated; **Inactive** — if the responsible for the document template deactivates it. Figure 6 presents a state machine diagram for a document template's workflow.

4.3. qDocs/Citizen

The qDocs/Citizen application is aimed at any citizen looking for the benefits of the system — allows citizens to access and manage their documents in a secure manner.

Figure 7 presents a UML class diagram of the domain model related to the qDocs/Citizen application.

Document represents a document that is accessible by a citizen.

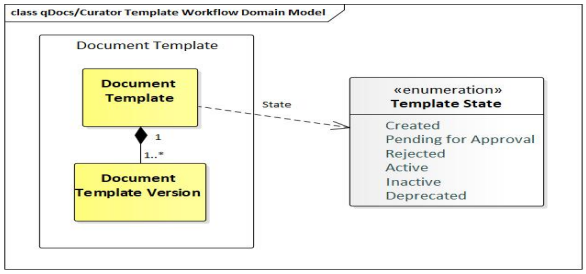

Figure 5: Document Template Workflow Domain Model(UML Class Diagram)

User represents all citizens using the qDocs platform. A user may be associated with documents as two different entities: owner or participant. If the user is a owner, then it may have multiple documents and a document can only have one owner. If the user is a participant, then it can be associated with multiple documents and vice-versa.

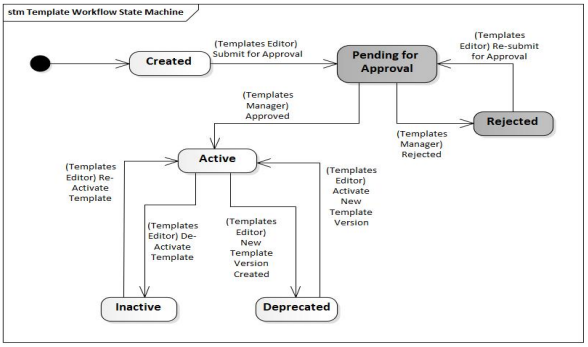

Figure 6: Document Template Workflow State Machine Diagram

Document Share represents all the information regarding a document that was shared, i.e. if a document is shared, it has information regarding who shared the document and who accessed it besides the owner of the document.

Curator represents the curator to which a document is associated. A document may be associated with one curator but a curator may be associated with multiple documents.

Document Types is comprised as the document groups to which a document is associated (e.g. forms or certificates). A document may be associated with multiple document types and vice-versa.

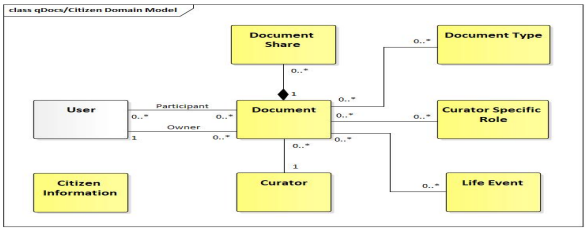

Figure 7: qDocs/Citizen Domain Model (UML Class Diagram)

Life Events is comprised as the life events to which a document is associated (e.g. birth or marriage). A document may be associated with multiple document types and vice-versa.

Curator Specific Roles (also known as user group) is comprised as the roles to which a document is associated (e.g. student or president). A document may be associated with multiple user groups and vice-versa.

Citizen Information represents all the information regarding a citizen, e.g. id, name and curators to which a citizen is associated with. This is a singleton class.

4.4. Cross-Cutting Concerns

During the development of the qDocs platform there are some requirements that are cross-cutting to the whole qDocs platform. The platform must provide security features such as confidentiality, integrity, authenticity, and non-repudiation. It should provide security at communication-level using a

Kerberos-based protocol; support Personal Data Privacy; support User Authentication based on citizen id and password or other means such as AMA's autenticacao.gov.pt service; and support User Authorization based on general and specific roles. Integration is another important concern, since qDocs is a cloud-based platform must support interoperability with other external system that will request information from different sources, some of them Curators with legacy applications [7, 8]. The platform must integrate with AMA's managed external systems. This will be done with qBox, which is a Curator' specific external system to provide such integration. Finally, the platform must support multi-language and take User Interface (UI) aspects, i.e the platform must be provide an attractive, simple and easy to use user experience.

In this work the focus is on the User Authorization based on general and specific roles and everything related with the User Interface (UI).

5. qDocs Technical Description

This section presents what is accomplished with this work. Sub-section 5.1 presents the architecture of the platform, sub-sections 5.2, 5.3 and 5.4 describe what is developed in each qDocs application and sub-section 5.5 describes the implementations that are common to the whole application.

5.1. Architecture Overview

Since it is a web based solution, the architecture is the one of a full stack application. The application has three major components: client, server and database. Users access the application through the client and make requests to the server through it. The server then retrieves the requested information from the database and sends it to the client. The database uses PostgreSQL[12], which is open-source object-relational database system that uses and extends the SQL language. The server uses the ASP.NET CORE framework and the client uses the Angular 4 framework.

5.2. qDocs/Admin

The qDocs/Admin application provides administration features for the qDocs ecosystem. It was already possible to create new curators and edit their information before this work, but it was not possible to edit all information regarding a curator, so these minor changes were developed. Also the settings interface was implemented with this work.

5.3. qDocs/Curator

The qDocs/Curator application allows curators to design and configure their document templates and make them available for citizens. Before this work, a user from the curator could: Associate users to the curator; Manage users associated to the curator; Manage curator-level entities; Manage data services; Manage form objects; Manage document templates; Design document templates, using form objects and snippets.

With this work, some features were enhanced (e.g. all managing interfaces have pagination and search bars) and others were created in order to provide a better experience for the users of the curator (e.g. associating users in bulk and configure features for document templates). These implementations include: Creation of curator General Roles; Settings interface development; Bulk association of users to the curator; Item deletion option (e.g. a Administrator could not delete a Form Object Group he created); Participant configuration; Template feature configuration; Search bars, pagination and filters to improve item search; Buttons to improve navigation between interfaces.

5.4. qDocs/Citizen

The qDocs/Citizen application is aimed at any citizen looking for the benefits of the system. Before this work, a citizen user could: Create documents; Consult its documents; Use the share document feature; Access and edit its profile information.

The purpose of this work regarding the qDocs/Citizen application was to test and validate what was implemented in the qDocs/Curator application, e.g. the participant configuration and document feature availability.

5.5. Cross-Cutting

During this work, all interfaces newly created kept the visual style that was already implemented before this work. In addition the following cross-cutting technical implementations were made: **Search Bars** and **Pagination** — all list interfaces did not have these features; they make the search for items in a list easier; **Navigation** — in some interfaces navigation was not flexible or not properly done, e.g. it was not possible to navigate from the template edit metadata interface to the template edit content interface directly and vice-versa; **Curator Name** — before this work, all curator interfaces did not have the respective curator's name on them; this was implemented to provide context to the curator user; **Authorization** — before this work, when a citizen was associated to a curator it was allowed to access the curator's interface, however it could not do anything.

6. Evaluation

Two scenarios were used to test the implementations done in the qDocs platform. Both scenarios present the life-cycle of a document in a fictional

[12] https://www.postgresql.org/about/, last accessed 30th July 2019

academic context. The first scenario (Scenario – A) is focused on the following aspects: Template metadata configuration, manually use of payment and strong authentication features; Access to a created document through different filtering options; Access to the created document and check the available features. The second scenario (Scenario – B) is focused on the following aspects of the qDocs application: Template design; Participant configuration; Document creation; Access to the created document; Participant invitation; Access to the document by a participant.

6.1. Discussion

The scenarios used were tested in a controlled environment, supervised by both advisors of this work, which are participants in the qDocs project. As stated, both scenarios presented the life-cycle of documents in the academic context. The first scenario was focused on the configuration of a template and its consequences for a citizen who wants to access a document created with that template. The second scenario was focused on the design of a template and its participant configuration. This scenario included the interaction of a citizen with a document created with the designed template and the interaction of the invited participants with that same document.

After this work, qDocs provides more functionalities to all kinds of users of the platform (administrators, curators and citizens): administrators now manage users, curators and services available for the curators; curator users have more options to configure their templates and the different curator general roles provide better management flexibility; citizens now can access shared documents as participants and have specific features available when accessing their documents.

On the curator side, qDocs gathers information from associated curators through the qBox application, allowing the platform to integrate with entities that use different systems. Regarding template definition, although it is not possible to import documents, qDocs provides an HTML text editor within the qDocs/Curator application, making template definition possible through any device with internet connection. With the definition of Data Services and Form Objects it is possible to gather (and store) data directly from curators' databases and use it in different templates, so there is no need to define a Form Object or Data Service twice for the same purpose.

On the citizen side, the process of document creation is similar in every platform, because it is a simple, user-friendly method. With this project, qDocs now has the participant configuration and invitation feature which is different from the platforms evaluated. It is scheduled for future work that citizens must make a request for document creation, and when the request is approved by a Documents Manager they may have access to the document. Regarding digital signature, it is scheduled for future work.

Regarding content management, which is common to both citizens and curators, it is done within the qDocs platform. When a citizen requests the generation of a document the information needed to do it comes directly from qBox, which is stored in the curator's system. This makes document generation secure, because the information stored in qDocs is relative to the information it needs to request to qBox, so sensitive information is never stored directly in the qDocs platform. Also qDocs is a cloud platform, so all its features are available to users by accessing the qDocs website.

Concluding, if qDocs is evaluated according to the framework defined in Section 2 it will have its values oriented to a citizen-centered platform as it should be. Comparing it to A2J Author, which is the other citizen-centered platform evaluated, qDocs may support a broader range of users, since every curator may join the platform, meaning every kind of document will be accessible to citizens.

7. Conclusions

Document automation is an approach that supports the creation of electronic documents in a flexible and efficient way, which provides several benefits for organizations. Document automation platforms are popular in several areas and keep evolving over time. However, these platforms are most common designed to be used by private organizations, therefore not addressing the purpose to help citizens in general. qDocs is a platform designed for that purpose, it provides a single point of access for citizens to access and manage their own documents (e.g. id cards, forms, certificates). These documents are provided by curators that use the qDocs platform to both manage templates and data. This work proposes an improvement to the qDocs platform, so that every citizen and curator may benefit with the use of the platform.

After analyzing the state-of-art, no other platform seems to match what qDocs offers. Being able to connect with any kind of curator, which has specific requirements regarding documents would be a major benefit for citizens because they would have a single point of access to all their documents. This work was evaluated in a controlled environment by applying qDocs to two scenarios that had different requirements.

7.1. Future Work

This sub-section describes a few implementations to the be done in the future to this work. These

suggestions emerged during the development of this work but due to reasons as time constraints or complexity they were not considered. The suggestions for future work are the following: **Curator General Roles Use Cases** — the Templates Manager, Documents Manager and Auditor curator general roles exist in the qDocs platform but they do not do yet anything; **Strong Authentication** — this feature was scheduled for future work due to its complexity of implementation and since many curators have documents that require citizen authentication in order to be accessed; with this work only the general logic regarding this feature was considered; **Payment** — this feature was scheduled for future work due to its complexity of implementation and since many curators have documents that need to be payed in order to be accessed; with this work only the general logic regarding this feature was considered; **Participant Notification** — when a user invites other users to have access and participate in his document, they should be notified with an access key or hyperlink automatically; **Document Manipulation by Participants** — participants can only access a document and see buttons relative to the features available for them; however these features should be implemented (such as Sign and Export) and interactions with the document like filling specific fields should be implemented as well.

References

[1] K. Betts and K. Jaep. The Dawn of Fully Automated Contract Drafting: Machine Learning Breathes New Life Into a Decades-Old Promise. *Duke Law & Technology Review*, 15(1):216–233, 2017.

[2] N. Colineau, C. Paris, and K. V. Linden. Automatically generating citizen-focused brochures for public administration. *ACM International Conference Proceeding Series*, pages 10–19, 2011.

[3] R. Eito-Brun and A. Amescua-Seco. Automation of Quality Reports in the Aerospace Industry. *IEEE Transactions on Professional Communication*, 61(2):166–177, 2018.

[4] J. Frank. A2J Author, Legal Aid Organizations, and Courts: Bridging the Civil Justice Gap Using Document Assembly. *Western New England Law Review*, 39(2), 2017.

[5] R. J. Glushko and T. McGrath. Document Engineering for e-Business. *Proceedings of the 2002 ACM Symposium on Document Engineering*, pages 42–48, 2002.

[6] T. F. Gordon. A Theory Construction Approach to Legal Document Assembly. *Expert Systems in Law*, pages 211–225, 1992.

[7] W. He and L. D. Xu. Integration of distributed enterprise applications: A survey. *IEEE Transactions on Industrial Informatics*, 10(1):35–42, 2014.

[8] R. Iqbal, N. Shah, A. James, and T. Cichowicz. Integration, optimization and usability of enterprise applications. *Journal of Network and Computer Applications*, 36(6):1480–1488, 2013.

[9] A. Kamphuis. Revolutionary Technology. *Courts Today*, 2012.

[10] M. E. Key. The Universalization of Data Interaction Technology Research between Database and Spreadsheet of WORD Based on OLE Qing-zhong JIA and Bin XIAO. *DEStech Transactions on Environment, Energy and Earth Sciences (SEEIE 2016)*, 2016.

[11] M. Kilger. A shadow handler in a video-based real-time traffic monitoring system. *Proceedings of IEEE Workshop on Applications of Computer Vision*, 1992-Novem:11–18, 1992.

[12] R. Klempner. The case for court-based document assembly programs: A review of the New York state court system's "DIY" forms. *Fordham Urban Law Journal*, 41(4):1189–1226, 2015.

[13] R. Lankester. Implementing Document Automation: Benefits and Considerations for the Knowledge Professional. *Legal Information Management*, 18(2):93–97, 2018.

[14] M. Lauritsen. Knowing documents. *Proceedings of the International Conference on Artificial Intelligence and Law*, Part F1271:184–191, 1993.

[15] M. Lehtonen, R. Petit, O. Heinonen, and G. Lindén. A Dynamic User Interface for Document Assembly. *Proceedings of the 2002 ACM Symposium on Document Engineering*, pages 134–141, 2002.

[16] N. Loutas, F. Narducci, A. Ojo, M. Palmonari, C. Paris, and G. Semeraro. PEGOV 2014: 2nd international workshop on personalization in eGovernment services and applications. *CEUR Workshop Proceedings*, 1181:1–9, 2014.

[17] MDSS. qDocs , Citizen centric document technology. *White Paper*, 2018.

[18] J. A. Menezes, A. Silva, Rodrigues da Silva, and J. Saraiva. Citizen-Centric and Multi-Curator Document Automation Platform : the Curator Perspective. In *Proceedings of ISD'2019, AIS*, 2019.

[19] A. Mesbah and A. V. Deursen. Migrating Multi-page Web Applications to Single-page A JAX Interfaces. *Mesbah, Ali, and Arie Van Deursen. "Migrating multi-page web applications to single-page Ajax interfaces." 11th European Conference on Software Maintenance and Reengineering (CSMR'07). IEEE*, 2007.

[20] V. Mital and A. D. Elliman. Document assembly and evidence analysis - Two approaches to hypertext, 1994.

[21] F. Monteiro. *Learning Single-page Web Application Development*. Packt publ edition, 2014.

[22] D. R. Mountain. Disrupting conventional law firm business models using document assembly. *International Journal of Law and Information Technology*, 15(2):170–191, 2007.

[23] S. Passera, H. Haapio, and M. Curtotti. Making the Meaning of Contracts Visible – Automating Contract Visualization. *IRIS Conference on Legal Informatics 2014, in Salzburg*, 450(February):443–450, 2014.

[24] M. C. Penadés, P. Martí, J. H. Canós, A. Gómez, M. C. Penadés, P. Martí, J. H. Canós, A. Gómez, P. Line-based, N. Loutas, F. Narducci, A. Ojo, and M. Pal. Product Line-based customization of e-Government documents. *In PEGOV 2014: Personalization in e-Government Services, Data and Applications (Vol. 1181). CEUR-WS*, 2014.

[25] N. J. Petro Jr. DOCUMENT AUTOMATION: Using Technology to Improve Your Practice. *GPSolo*, 32(5):56–62, 2015.

[26] R. Smith and A. Paterson. Face to Face Legal Services and their Alternatives : Global Lessons from the Digital Revolution. *White Report*, 2014.

[27] M. W. Wong, H. Haapio, S. Deckers, and S. Dhir. Computational Contract Collaboration and Construction. *Proceedings of the 18th International Legal Informatics Symposium IRIS 2015.*, 512(February):505–512, 2015.