READ
RECOGNITION & ENRICHMENT
OF ARCHIVAL DOCUMENTS

D4.2
READ Platform and Service Maintenance

Philip Kahle, Sebastian Colutto, Günter Hackl, Günter Mühlberger
UIBK

Distribution: http://read.transkribus.eu/

READ
H2020 Project 674943

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 674943

Project ref no.	H2020 674943
Project acronym	READ
Project full title	Recognition and Enrichment of Archival Documents
Instrument	H2020-EINFRA-2015-1
Thematic priority	EINFRA-9-2015 - e-Infrastructures for virtual research environments (VRE)
Start date/duration	01 January 2016 / 42 Months

Distribution	Public
Contract. date of delivery	31.12.2017
Actual date of delivery	23.11.2017
Date of last update	08.12.2017
Deliverable number	D4.2
Deliverable title	READ Platform and Service Maintenance
Type	Demonstrator
Status & version	FINAL
Contributing WP(s)	WP4
Responsible beneficiary	UIBK
Other contributors	All partners
Internal reviewers	Gundram Leifert, Hervé Dejean
Author(s)	Philip Kahle, Sebastian Colutto, Günter Hackl, Günter Mühlberger
EC project officer	Martin Majek
Keywords	Transkribus

Contents

1 Executive Summary

This deliverable outlines the progress of task 4.1, READ platform and service maintenance, which involves activities such as updating background systems, bug and error handling, system migration (to larger servers according to the expanding network), user support, and similar activities.
In the first year of the project, activities were focussed on improving the overall user experience by fixing existing bugs in the system and extending functionality. On the other hand, the architecture of the platform was overhauled with a focus on scalability which eases the addition of hardware resources in the future. Furthermore, a migration of parts of the system to the University of Innsbruck computing center aimed at providing improved availability of the platform.
In the second year the main goal in this task was to improve existing functionality; only urgently needed features have been added. Also the overall architecture designed in the first year has proven to be robust and was just slightly adapted where shortcomings were revealed. Due to this, the platform could be scaled hardware-wise as planned.
This deliverable is divided into three parts: the first deals with service maintenance, the second part describes the adjustments in the architecture of the platform, and the last part provides some details on currently used hardware resources and additions in the near future.

2 Service Maintenance

The starting point of the READ platform was the existing Transkribus system, developed in the TranScriptorium project. Transkribus allows a user to ingest sets of document images into the system, where they are stored persistently, transcribe and enhance them in a standardized way and, finally, export them in different formats, such as METS/ALTO, PDF, Word or Excel. Several integrated tools ease the transcription process with automated steps, e.g. finding regions and/or lines in the images or recognizing the text.
When the project started, Transkribus had a user count of 2828 (as of 1.1.2016). During the first year, the dissemination activities led to 2082 new users that could be acquired (28.11.2016). Between that date and 24.11.2017 3436 registered in the platform which is a significant increase compared to the first year. UIBK received about 272 feature requests and bug reports in 2016 (until 28.11.2016) and since then 306 of those[1]. Putting that in correlation to the number of new user registrations, one can reason that the base functionality used by most users is less prone to error.
In 2016, the platform was unavailable for 7 hours due to scheduled maintenance. One incident with a malfunctioning storage system caused an unscheduled downtime of 17 hours; an issue which will be reflected in the migration strategies in 2017. The service availability for the first year can thus be stated with 99,73%. Due to the modular architecture (see section 3)it was easier to update parts of the platform and therefore the scheduled downtime could be reduced to 3,5 hours in 2017. However there still occured problems that caused unscheduled downtime or malfunctionioning of parts of the sys-

[1]The numbers include all reports that have been sent via the reporting feature in the Transkribus GUI

tem. As the monitoring solution in place did not cover all of those incidents, only an estimation of about 20 hours of unscheduled downtime can be given, which yields an overall availability of ∼99,73%.
Strategies to increase that number will be followed in the third year of the project: while some sources of error are now known and may therefore be eliminated it is crucial to put more effort into server administration and testing to counter the unexpected. As mentioned before, UIBK maintains a monitoring system that can help to detect problems earlier if configured exhaustively. The same goes for UIBK's build server[2] that automatically runs software tests and can notify developers about potential problems before they disrupt operation.

2.1 Software Development in Year Two

The second year started with an important milestone: Transkribus GUI version 1.0 was released, marking the tool, more or less, as feature-complete. Since then only a few urgently requested features and those related to the integration of tools (see D4.5) have been added to the client application. The main focus was set on improving existing functionality and the back-end services.
While there are still often feature requests that may very well be useful for specific workflows it has to be stressed that the complexity of the Transkribus platform has reached a level that is hard to manage for a small development team. Besides the tests done on a new feature it has to be thoroughly tested whether existing functionality is broken by the changes. In case a fitting test does not exist yet this code has to be written too. Moreover, for each existing feature there will be at least one reasonable request by a user to improve it in a certain way. Therefore, each new feature must be evaluated carefully in the third year and there are also plans to remove some that did not stand the test of time or are rarely used. Refactoring and enforcing the reuse of solutions that have proven to be robust and easy to use is also crucial to ease maintainability. The following reduction in complexity will then create space to add the tools that have not yet been integrated in year three.

3 Architecture

In year one, the server application of Transkribus was split into two parts: one web application (TranskribusServer) that serves the data to clients via its REST API and a worker application (TranskribusAppServer) responsible to handle heavy workload tasks such as layout analysis or text recognition. Moreover, the Quartz scheduler framework was used to implement queues for specific tasks and executing the jobs from those queues. As the number of different job tasks grew this year, it became clear that a more flexible system is needed to cater for the specific needs in the Transkribus platform. Thus, a custom-made solution was put into place which also allowed to further modularize the platform: with the current system it is possible to implement applications to handle specific job types or even just one of those, e.g. keyword spotting requests are now dealt

[2] https://jenkins.io/

with in an application including that specific module. Once the workload in a module reaches a critical level, it is possible to deploy and start more instances of the module; the system automatically distributes the workload among them. While the scalability was already vastly improved in year one, the modular approach allows to scale the system according to the current needs and available hardware. An overview of the platform architecture is shown in figure 1.

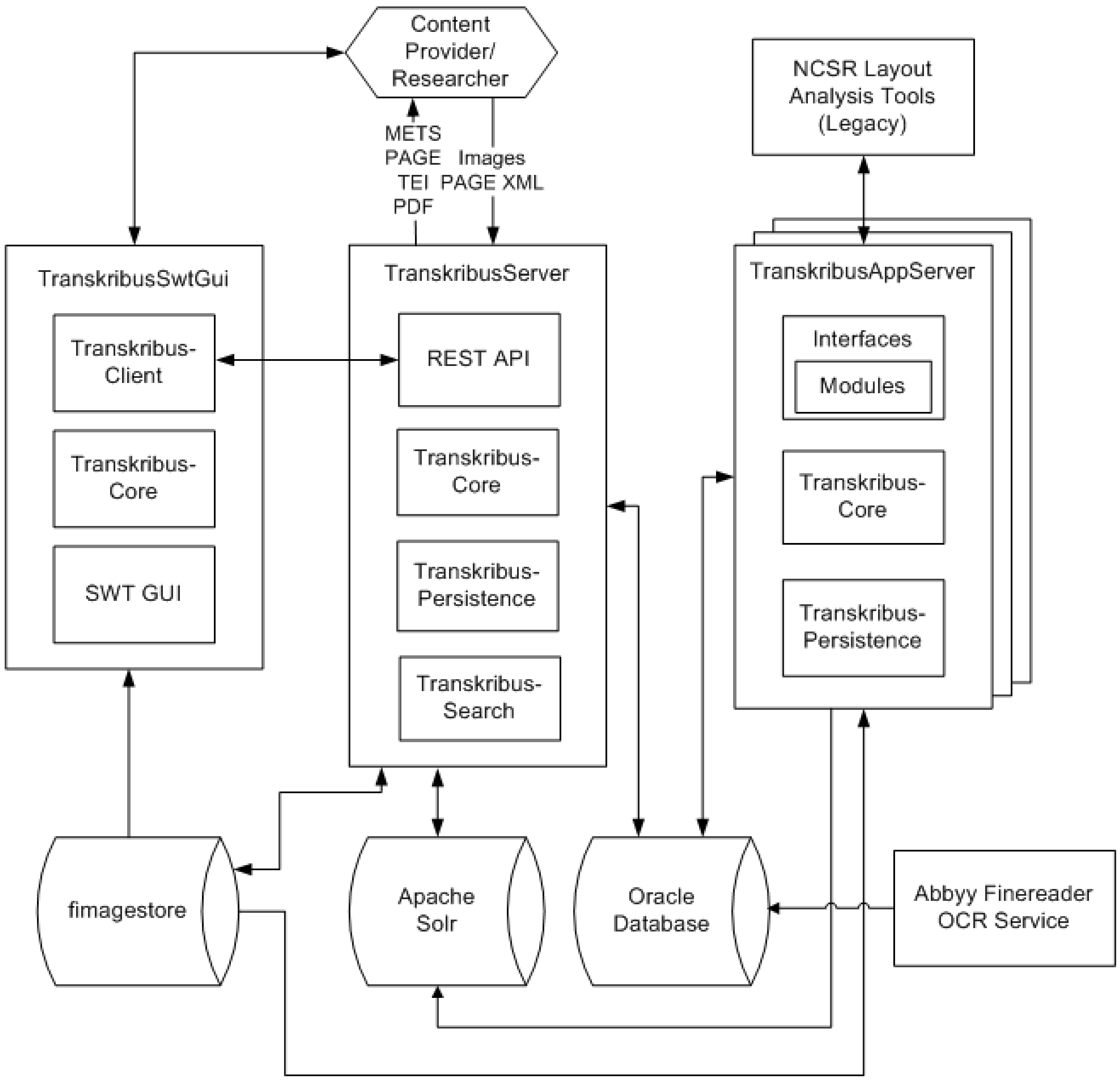

Figure 1: Current Transkribus architecture

3.1 Importance of the RESTful API

As described in section 2, a point in the project is reached where the maintenance of the code base of Transkribus becomes quite time consuming and features have to be chosen selectively before being implemented.
However, very specific requirements of transcription projects do not exclude using Transkribus for their purpose. The RESTful API of the system allows to interface the system to build necessary functionality outside of the platform. For instance, a project at the Austrian Academy of Sciences[3] uses a custom-built website, visualizing the progress of the project based on data from the system's API. Another important example is of course the Transkribus webinterface built in the project that uses this interface (see D5.6).
While this service was at first built with the Transkribus GUI in mind, the second year of the project showed that this API is a very valuable asset and future design decisions regarding the endpoints of the service will reflect this importance by keeping the entry threshold to the usage as low as possible.

4 Hardware

In the end of the first year, UIBK acquired an HP Bladecenter including 16 servers with 12 cores each. To the date of this writing, 9 of those machines have been added to the Transkribus platform, running instances of TranskribusAppServer. Only few load peaks have been experienced where the setup was exhausted in the second year.
The most expensive task is currently the training of the HTR engine (see D7.8) and in year three, the used engine by URO will be backed by the well-known Tensorflow framework, which allows to speed up the training phase using graphics processing units (GPU). In this turn, UIBK is planning to add respective server machines including GPUs to the platform, which promises a massive increase in efficiency. Taking into account the trend in the number of registered users (see section 2) the standby hardware should then be sufficient at least until the third quarter of 2018.

5 Conclusion and Outlook

While the effort put into this task could be reduced to some degree as expected, the increase of active users and also in the platform's code complexity still required a lot of attention. Maintaining high quality of service will involve an increased amount of resources that has to be put into service maintenance in the third year. Upcoming challenges include administrating the infrastructure, maintaining the code base of the platform and polishing existing functionality.

[3]https://www.oeaw.ac.at/en/acdh/projects/event-detail/article/wiennerisches-diarium-digital-pilotstudie-zur-erschliessung-einer-historischen-zeitung/ (project description in German, accessed on 27th of November 2017)