Grant Agreement No.: 761488

D3.4: Technology Bricks V3

This deliverable provides the technology bricks which have been planned for the third prototype of the CPN platform.

Work package	WP 3
Task	
Due date	31/10/2019
Submission date	27/12/2019
Deliverable lead	ATC
Version	Final
Authors	Thomas Sounapoglou, Eva Jaho (ATC)
Reviewers	Ferdinando Bosco (ENG)
Keywords	Technology Bricks, APIs, prototype 3

Document Revision History

Version	**Date**	**Description of change**	**List of contributor(s)**
V0.1	15/10/2019	Table of Contents	Thomas Sounapoglou (ATC)
V0.w	29/10/2019	Initial version	Thomas Sounapoglou, Eva Jaho (ATC)
V0.3	10/12/2019	First completed version	Thomas Sounapoglou, Eva Jaho (ATC), Ferdinando Bosco (ENG), Matthias Strobbe, Chris Develder, Thomas Demeester, Johannes Deleu (IMEC), Fulvio D'Antonio (LIVETECH), Robert Learney, Daniel Puschmann, Anthony Garcia (DCAT),
V0.4	16/12/2019	Final edits	Thomas Sounapoglou (ATC), Ferdinando Bosco (ENG)
V0.5	20/12/2019	Reviewed version	Ferdinando Bosco (ENG)
V1.0	27/12/2019	Completed version incorporating all comments received	Thomas Sounapoglou (ATC)

DISCLAIMER

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 761488.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 761488.

This document reflects only the authors’ views and the Commission is not responsible for any use that may be made of the information it contains.

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:	**R**	
Dissemination Level		
PU	Public, fully open, e.g. web	**X**
CL	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to CPN project and Commission Services	

EXECUTIVE SUMMARY

This deliverable reports on the work performed in WP3, which addresses the development of the required technology bricks for the CPN platform. The input to this document were the deliverables D1.1 "User Requirements Model"[1] and D1.4 "Technical Requirements (platform and service requirements)"[2]. The scope of this document is to report the components and services implemented for the third prototype of the platform. As defined by the project's requirements, these components and services are classified in three categories: Content, Users, Mapping.

For each technology brick, a brief description of its functionality is provided, along with existing API, test scenarios and installation guidelines.

[1] https://www.projectcpn.eu/s/CPN_D11_User-Requirements_201802028_V10-2pkf.pdf

[2] https://www.projectcpn.eu/s/CPN_D14_Technical_requirements_platform_and_service_requirements_20180830_v10.pdf

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

API	Application Programming Interface
ATC	Athens Technology Center
CPN	Content Personalization Network
DCAT	Digital Catapult
DW	Deutsche Welle
ENG	Engineering Ingegneria Informatica
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
IMEC	Interuniversity MicroElectronics Center
JSON	JavaScript Object Notation
JWT	JSON Web Token

NLP	Natural Language Processing
RDF	Resource Description Framework
REST	Representational State Transfer
RSS	RDF Site summary
UI	User interface
UR	User requirement
VRT	Vlaamse Radioen Televisieomroep
YAML	YAML Ain't Markup Language

1. INTRODUCTION

This Deliverable contains the basic description of the technological infrastructure of the third prototype of the CPN platform, which is composed by what we call 'technology bricks'. The components, APIs, and services included in the third version of the platform customization infrastructure and components, and described in this deliverable have been designed and developed according to the user requirements, as described in deliverable D1.1 “User Requirements Model”.

The CPN project foresees three releases of the 'technology bricks' in order to be available for the related pilots. Each release includes specific functionalities, chosen after a process of evaluation and prioritization of the user requirements. Based on the reference architecture document (D2.1)[3] and the second version of the technology bricks deliverable (D3.3)[4], the third version of the technology bricks is the third and the final cycle of three iterations and will offer a series of features in order to test the related bricks in a pilot (third Pilot) environment.

The main goal of this document is to present the technology bricks that are foreseen at this point of the project necessary to satisfy the user requirements expected for the third pilot iteration. For each technology brick, a brief description of its functionality is provided, along with any existing APIs, test scenarios and installation guidelines. In addition to the new technology bricks implemented, this deliverable also describes the updates of the already deployed technology bricks, highlighting new features implemented and how these bricks satisfy the user and technical requirements.
For each technology brick described in this document, a follow up of the comments received from the Reviewers during the second Review of the project is provided.

The structure of the deliverable is organized as follows: Section 2 provides an overview of the requirements for the third prototype, Section 3 describes the new CPN technology bricks that are introduced for the 3rd prototype of the platform, Section 4 provides updates of the already available technology bricks, and Section concludes this document.

[3] https://www.projectcpn.eu/s/D21-CPN-Reference-Architecture-v10.pdf
[4] https://www.projectcpn.eu/s/D33-Technology-Bricks-V2.pdf

2. SUMMARY OF REQUIREMENTS FOR PROTOTYPE 3

The requirements foreseen for the third prototype along with the list of the modules that have been necessary for satisfying these requirements are enlisted below.

Note that some requirements are discarded according to document D6.5 2nd Review Periodic Report.

Requirement category	Requirement ID	Requirement Description	Notes
UR-UP 1: Interests (Categories, Entities, Values): What topics is the user interested in?	UR-UP 1.3	The system should be able to offer personalized content on the basis of the users mood or values	
The system should be able to offer personalized content on the basis of the users mood or values	UR-UP 4.1	The system must allow the user to choose preferred types of content	Discarded: Having different media types has been evaluated too technically complicated, so we decided to focus on the text first.
	UR-UP 4.2	The system should set/refine preferred types of content based on the user's consumption habits and the timing	
	UR-UP 4.3	The system should refine the user's preferred types of content through frequent interaction with	

		the user (talkback)	
UR-UP 6: Knowledge (Management): What does the user already know?	UR-UP 6.4	The system should be able to offer insights and advice based on what it learn about what a user consumed in relation to a certain entity (e.g. a place)	
	UR-UP 6.5	The system should allow the user to delete part of the systems knowledge for specific time frames back in time from the moment of viewing	
UR-UP 7: Devices: On what device is the user consuming content?	UR-UP 7.1	The system should check on what device the user is consuming the content	
	UR-UP 7.2	The system should adjust its content offering based on the type of device the user is using	
	UR-UP 7.3	The system should try to make smart use of device data to determine the surroundings of the user and	Discarded: Using the device's sensor is out of scope

		adjust the content strategy accordingly	
UR-UP 8: Importance for user: What is relevant for the user, outside their given interests?	UR-UP 8.1	The system should combine reading habits and knowledge about the user to provide smart updates on things the user could be interested in, even if this doesn't fit his/her set interests	Discarded: This user requirement has been discarded due to the lack of data collected about the users to enrich their profiling. Social media networks become huge private marketplaces and only twitter continues to be open and transparent.
	UR-UP 8.3	The system should be able to surprise the user with content, he/she would not have chosen themselves	
UR-UP 9: User Profile Management: Giving the user transparency and control over their data	UR-UP 9.6	The system should allow the user to add external data to update their profile	
UR-AF 1: Bursting the Filter Bubble: How can CPN avoid filter bubbles and echo chambers?	UR-AF 1.1	The system should offer users an overview of other sources, covering the same topic	Discarded: This was deemed unnecessary as every user is attached to a single source
	UR-AF 1.3	The system should offer the user an easy overview of what content from which	

		sources he has consumed over a certain period of time	
UR-AF 3: Content/Format: In which way do we have to prepare content for the user?	UR-AF 3.2	The system should offer the user a short overview of all important headlines at a specific point in time with access to more details upon request	
UR-AF 5: Transparency: Giving the user control & understanding over the content he sees	UR-AF 5.3	The system must make it transparent to the users why they are shown certain content, based on an item level	
UR-AF 6: Archive: Making content available beyond the moment	UR-AF 6.4	The system should be able to memorize where a user left off and restart at the same point	
UR-AF 7: User Feedback: Asking users to help improve the system	UR-AF 7.1	The system should offer user feedback requests in a playful/entertaining way	
	UR-AF 7.3	The system should allow users to assign both existing or new attributes (categories, moods etc.) to a content item	

	UR-AF 7.4	The system should be able to offer a feedback interaction to determine the ground level of personalization based on mood, time and interest	
UR-PS 1: Detailed Analytics: Giving Newsrooms a more detailed feedback on their audience	UR-PS 1.4	The system should be able to show these numbers during the creation process of the content	
UR-PS 2: Integration: How should CPN be connected to the production side?	UR-PS 2.1	The system should allow for an easy integration into the producers workflow	
	UR-PS 2.2	The system should provide contract templates to allow freelancers to easily work together and with editors, to define and track the scope of individual contributions and expected revenues	
	UR-PS 2.3	The system should allow producers to transparently see how often their	

		contributions are used and distributed to readers	
	UR-PS 2.4	The system should allow producers to export the record of their publications through standardized and interoperable formats	
	UR-PS 2.5	The system should allow for an easy contribution of content from different publishers through standardized interfaces	Discarded: In agreement with user media partners, content editing within the dashboard in no longer expected. They prefer to have control over the editing in their environment
	UR-PS 2.7	The system should allow editors to easily add missing attributes to articles manually	Discarded: In agreement with user media partners, content editing within the dashboard in no longer expected. They prefer to have control over the editing in their environment

Table 1: List of Requirements for the third Prototype

The third prototype of the CPN platform introduces two new bricks and updates seven more as follows:

Layer	Name	Status
Content Technology Bricks	Fine Grained Entity Recognition Module	New
	Topic Extractor	Updated
	Recommender AB-Testing	Updated
User Technology Bricks	User Modelling	Updated
	Reader’s App	Updated
	Personal Data Receipt	Updated
Mapping technology Bricks	Distribution Framework	New
	Producer’s App	Updated
	Recommender	Updated

Table 2: Technology Bricks for Prototype 3

3.1.3 Internal Architecture

This section provides a brief overview of the information extraction model and its global neural network architecture. More details on the model, multiple alternatives, and details on the different components can be found in the technical report.

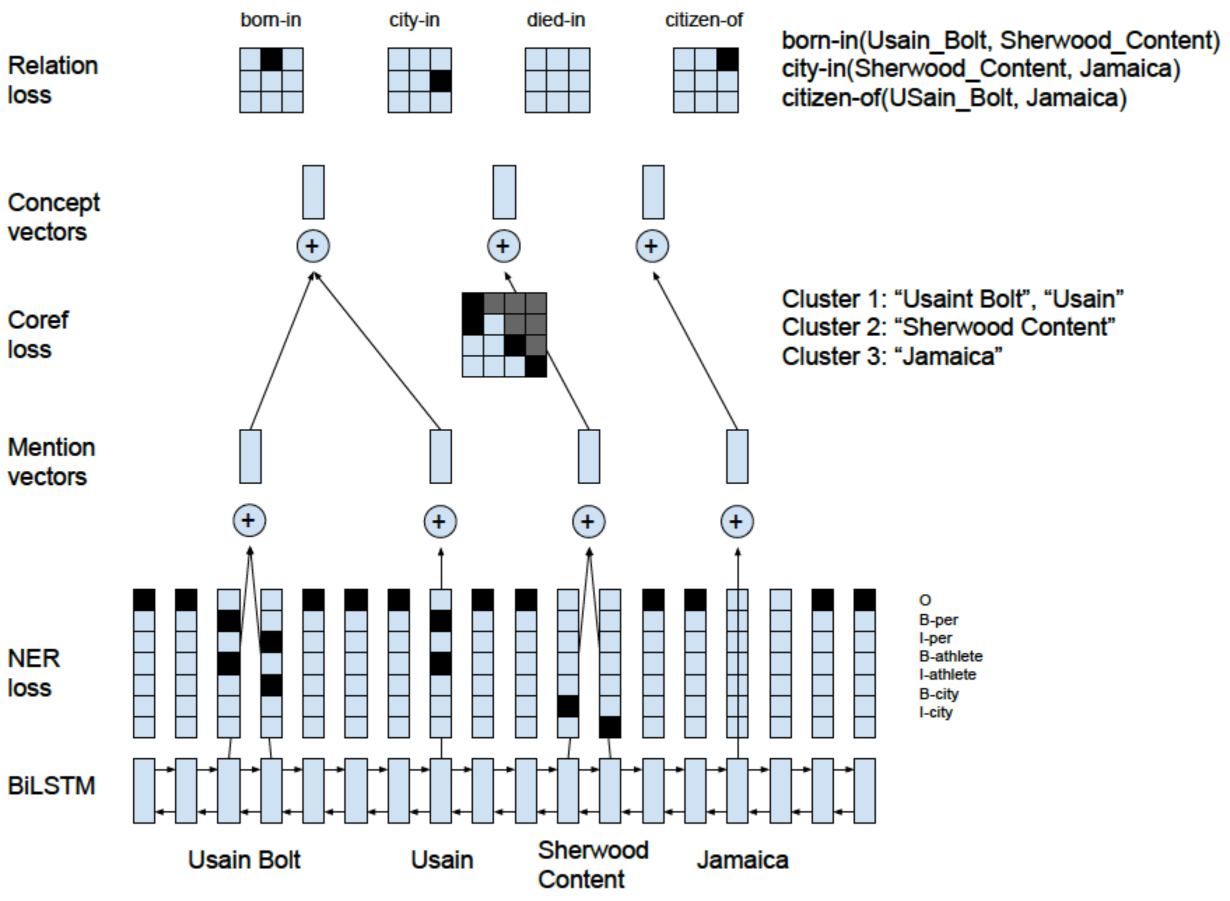


Figure 1: Schematic Overview of the Fine Grained Entity Recognition Module

A schematic overview is shown in the figure above. The model is organized as follows (see figure, from bottom to top):

- The token representation component (BiLSTM in figure), a bidirectional LSTM layer, reads in the text and generates a contextual representation for each token.
- The entity prediction component (NER loss in figure) uses this representation to decide where named entities are located in the text and what their types are.
- The mention representation component (mention vectors in figure) uses the predictions of the previous component to create a single representation per entity span.
- The entity clustering component (coref loss in figure) uses the mention representations to predict which mentions are clustered together, as they denote the same entity
- The entity representation component (concept vectors in figure) uses the predictions of the entity clustering component to generate a single entity vector per cluster
- The relation prediction component (relation loss in figure) decides for all pairs of entities which of the predefined relations -- i.e., none, one, or more -- they are engaged in.

3. NEW MODULES DESCRIPTION

This section describes the modules needed to implement the third prototype of the CPN platform.

3.1 Fine Grained Entity Recognition Module

3.1.1 Overview

The fine-grained entity recognition module involved building a dataset, designing information extraction algorithms, and actually training and evaluating models to automatically enrich news articles.
This document gives an overview of the research for this module performed at the IDLab research group of Ghent University - imec, and is accompanied by an extensive technical report. That report provides an overview of the newly created dataset, its unique properties, the models developed for various information extraction tasks on the dataset, as well as experimental results. In parallel, scientific publications are being prepared on the dataset as well as the new models, to be submitted early 2020.

3.1.2 Role

The goal of the fine-grained entity recognition module is information extraction on news articles. Within the CPN project, the extracted facts (fine-grained named entities, as well as relations among them) are available as structured features for enhancing the content-based part of the recommendation engine. Besides that, they could support a better navigation through the data or be used for visualization purposes.

The research performed for this task involved (i) building a new annotated dataset with English content from Deutsche Welle, and (ii) designing suitable information extraction models for it. The dataset has a number of unique properties that set it apart from existing information extraction datasets. For example, it is a random sample from an actual online news corpus (rather than a manually crafted composition of articles for more academic information extraction research), and rather than focusing on the sentence level, it considers the entire document for defining named entities and relations between them. This opens up the potential for more robust information extraction, but comes with additional technical challenges.

3.1.4 API

The following API description applies to the proof-of-concept software available as a Docker image published on CPN private Docker registry.

Input data

To process a news article, a request is sent to http://hostname:8080/process. The service expects two variables to be sent as form-data: text and format. Optionally an id variable can be specified for content identification purposes. Both HTTP GET and HTTP POST are allowed.

- The id variable (optional) can contain any string value. It will be copied back in the output.
- The text variable must contain the content of the article. It is recommended to concatenate title and body together, separated by a newline character.
- The format variable specified the requested output format. Allowed values are: html, simplified-json and json (advanced output).

Variables:

Output data

First an example of the simplified json format (format=simplified-json) is given. This format is recommended as it tries to summarize the more exhaustive advanced output. The output structure returns a list of keywords, the main countries involved, the main topics (of the entities) and the main entity types.

```
{
  "keywords": [
    "US",
    "Donald Trump",
    "Ukraine",
    "Joe Biden"
  ],
  "locations": {
     "US": 12,
     "Ukraine": 8
  },
  "topics": {
     "media": 1,
     "politics": 14
  },
  "types": {
     "entity": 17,
     "gpe": 2,
     "gpe0": 2,
     "head_of_state": 1,
     "igo": 1,
     "location": 2,
     "minister": 1,
     "organization": 7,
     "party": 1,
     "person": 8,
     "politician": 6,
     "politics_institution": 4,
     "politics_per": 2,
     "role": 2,
     "so": 1,
     "time": 2,
     "value": 5
   }
}
```

Next we show the full output (format=json) for the same news article. The output contains a copy of the input data (id and content), but also a list of recognized mentions, a list of concepts and a list of relations. Each mention refers to the concept it is associated with through its concept attribute (index in concept array). Concepts have the following attributes: text (the longest surface form), count (how many times they occur in the text), a list cpn types, a list of topics, slot types (if relevant), IPTC codes (if relevant) and a list of wikidata instance types (Q codes). The tag attribute must not be used and is included for debugging purposes only. Q codes can be looked up on wikipedia and wikidata, they enable you to filter on very specific entity types: e.g., Q1520223 refers to a constitutional republic. Relations are expressed as (subject, predicate, object) triples. Both subject and object are indices into the concept array.

```
{
  "id": null,
  "content": "A top US diplomat has told an impeachment inquiry that he followed President Donald Trump's orders to put pressure on Ukraine to investigate his Democratic rival, Joe Biden.\r\n\r\nThe instruction came from Mr Trump's personal lawyer, Rudy Giuliani, Ambassador Gordon Sondland said.\r\n\r\nThe inquiry is assessing if Mr Trump withheld military aid to Ukraine as a precondition. He denies any wrongdoing.\r\n\r\nIt is illegal in the US to seek foreign help to gain electoral advantage.\r\n\r\nMr Biden is one of the top contenders for the Democratic nomination for the 2020 presidential election.\r\n\r\nMr Sondland, the US ambassador to the EU, told the latest hearing in the US House of Representatives that Mr Giuliani had sought a public statement from Ukraine's leader, Volodymyr Zelensky, announcing an inquiry into \"corruption issues\".\r\n\r\nMr Giuliani specifically mentioned the company Burisma - which had the son of Democratic presidential candidate Mr Biden, Hunter, as a board member - and issues surrounding the 2016 US presidential election, he said.\r\n\r\nSondland's testimony and reaction\r\nFive key moments from impeachment hearing\r\nWho's who in Trump-Ukraine story?\r\nWhy Ukraine is so important to the US\r\nIf found guilty in a majority vote in the House, the Republican president will face an impeachment trial in the Senate. But two-thirds of members of that Republican-controlled chamber would then need to vote for Mr Trump to be removed from office.\r\n",
  "mentions": [
    {
    "begin": 6,
    "concept": 0,
    "end": 8,
    "text": "US"
    },
    {
    "begin": 67,
    "concept": 1,
    "end": 76,
    "text": "President"
    },
    ...
  ],
  "concepts": [
    {
    "text": "US",
    "count": 12,
    "type": [ "location", "entity", "gpe", "gpe0" ],
    "topic": [ "none" ],
    "slot": [ "keyword" ],
    "iptc": [],
    "auto": [ "Q1646605", "Q1520223", ... ],
    "tag": [ "type::location", "type::entity", "topic::none", "type::gpe", "slot::keyword", "type::gpe0" ],
    },
    ...
  ],
  "relations": [
    [ 21, "member_of", 4 ],
    [ 15, "institution_of", 0 ],
    [ 16, "institution_of", 0 ],
    [ 23, "institution_of", 0 ]
  ]
}
```

3.1.5 Testing Scenarios

The information extraction modules have been tested extensively on held-out articles from the new dataset with the appropriate metrics (F1 for named entities, B-cubed F1 for co-reference...). Below, some scientific conclusions are formulated in terms of neural network design for the dedicated information extraction tasks. For detailed numerical results, we refer to the technical report. Besides the extensive evaluation of information extraction quality, the impact of the extracted features on the recommendation quality has not been measured yet; this is currently under discussion for Pilot 3.

3.1.6 Conclusion and Overview of Research Findings

In summary, we created a new dataset to be used for a variety of information extraction tasks on an actual real-world news corpus, based on English content from Deutsche Welle. The annotation instructions were described extensively in a separate guidelines file. In accordance with the guidelines, a large number of annotations was performed, to be used for training and evaluating the information extraction tasks.

Related to a number of unique properties of the dataset, in terms of article-level annotations for entities and relations (today often done on the more limited sentence level), new neural network architectures were designed. These were evaluated extensively. The results were reported in detail, and as a proof-of-concept, a Docker container was delivered with access through a well-documented API to the overall best performing models.

We conclude with a short summary of the main scientific findings, based on the many different neural network architectures designed and tested for the information extraction tasks on the Deutsche Welle dataset:

- ➔ Adding a token aggregation layer, which shares information over different occurrences of a token in the entire article, results in better results for Named Entity Recognition (NER).
- ➔ Span based models for NER are able to compete with models using the traditional BIO encoding scheme.
- ➔ In comparison to NER, the co-reference and relation tasks prefer shallower recurrent neural networks (RNNs) and wider convolutional neural networks (CNNs) for processing n-grams.
- ➔ Iterative scoring functions clearly outperform non-iterative scoring functions (such as dot product, biaffine and feed forward networks).
- ➔ Propagation methods based on updating node representations in an iterative manner are not beating edge representation models, even with advanced gating functions.
- ➔ Our score propagation model beats all other iterative models, even though it is much simpler. This indicates that the models that update node representations are somehow missing something.
- ➔ Although the mention-wise latent relation loss is able to beat concept-wise relation prediction with non-iterative scoring functions, we were not able to achieve better results for the latent loss with propagation methods. A disadvantage is that they are not able to aggregate information from mentions associated with the same concept.
- ➔ Multi-task learning of the NER, co-reference and relation does not lead to better results yet. One problem is that individual models have slightly different optimal hyperparameters (e.g. co-reference is sensitive to higher dropout values, while other models require them). Another problem is that of selecting optimal task weights.

3.1.7 Installation and administration guidelines

The module is deployed as a Docker container into the CPN platform.

3.2 Distribution Framework

3.2.1 Overview

Current licensing and ownership models in journalism require lengthy negotiations to access, redistribute, and remix content. In the age of the 24hr news cycle this can make it difficult to source quality content from freelancers in a timely and secure manner, while also ensuring provenance.

The Distribution Framework aims to simplify the contractual negotiations between creators and editors by creating a pool of licensed content collaboratively managed by multiple trusted journalistic organizations without a single overarching authority. This is achieved by using Distributed Ledger Technology, in particular

Hyperledger Fabric, to manage the article licenses.

3.2.2 Role

The Distribution Framework serves to ingest journalistic content from media organizations and freelancers, enable transparent linkage to a human-readable license through a simple website, and present this to other organizations for sourcing and use according to the terms of the license.

The Framework sits between all participants in the system, preventing disagreement and simplifying the process of sourcing and distributing news content. In the future this could be extended with pricing and payments to create a trustless distributed content marketplace.

3.2.3 Internal Architecture

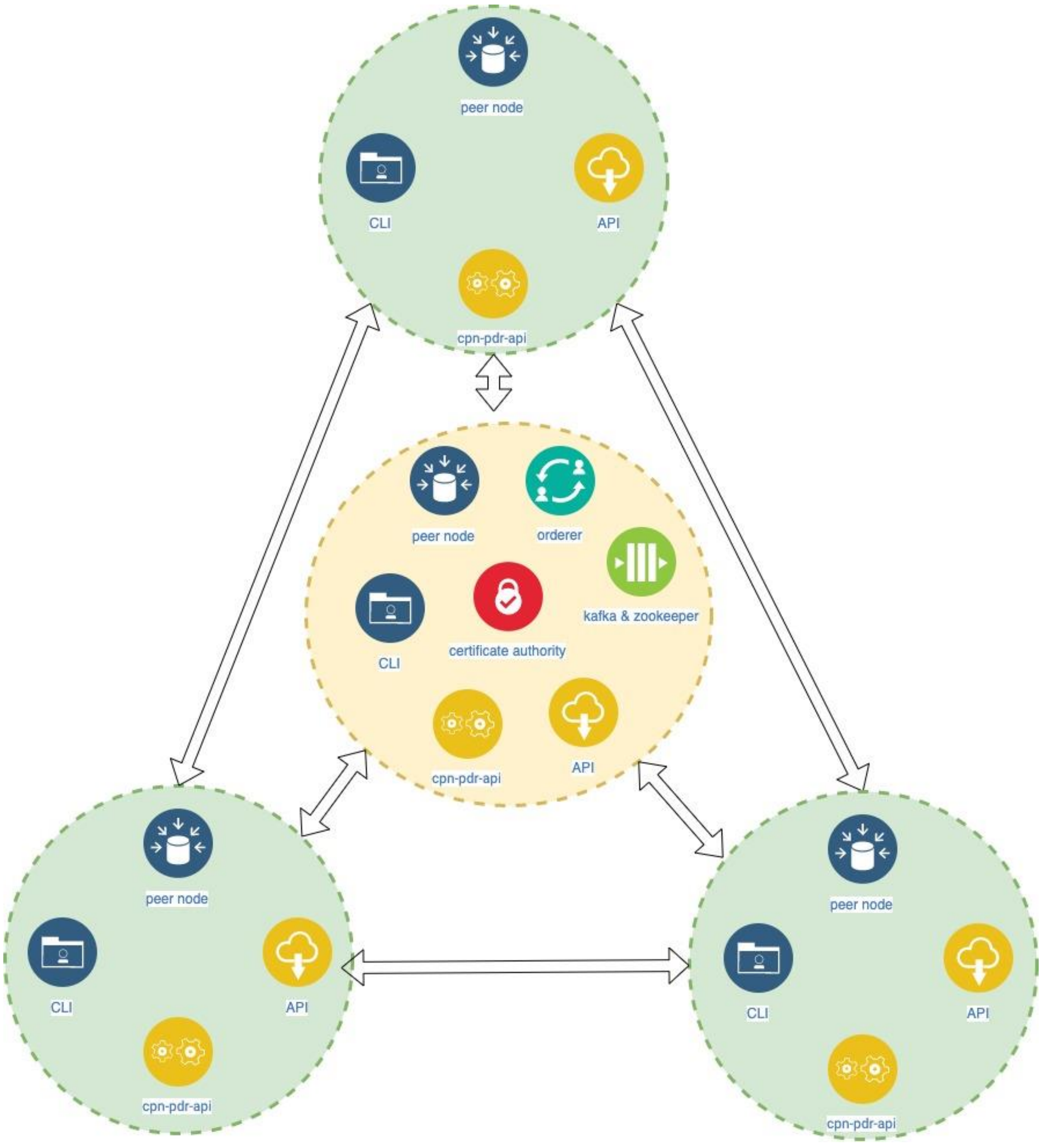


Figure 2: Architecture schema of the Distribution Framework module

3.2.4 APIs

The peer nodes expose two levels of APIs:

- An internal REST API, which directly interacts with assets on the chain.
- A public GraphQL API, which interacts with the REST API.

The REST API allows for basic CRUD operations on the different assets stored on the blockchain (content meta-data, licenses, agreements, ownership…). Changes to the chaincode will also affect the REST API. For this reason, we have built a pGraphQL API on top of the REST API. This way we can ensure not to break the CPN client applications that are storing and accessing assets from the blockchain.a JWT token is expected from the clients.

3.2.5 Testing Scenarios

The network was first deployed in a very limited configuration: one peer node, one ordering node, one certificate authority. This limited setup allowed for a rapid development and testing the chain-code (smart-contract), APIs and Frontend.

Once the chain-code and APIs reached a certain maturity, in terms of feature and stability, we switched our focus on the production network deployed on multiple nodes (see figure above).

The scenarios tested of the production network are:

- ➔ Ensure the nodes can communicate with each other's
- ➔ Ensure the network can still accept transactions when less than half of the peer nodes are down
- ➔ Ensure access control list are defined and enforced
- ➔ Ensure the API is not available without authentication

A dedicated section on the Producer's dashboard UI will be provided in order to allow to editors to exploit some of the distribution framework functionalities directly from the dashboard.

3.2.6 Installation and administration guidelines

All services used by the distribution framework are deployed as Docker containers, thus portable to any cloud providers or on premise servers that are Docker-compatible. To facilitate the administration of the distributed network, we provide a collection of bash scripts that allow for deployment of a new peer in the network, creation of the genesis block and installation/upgrade of the chain-code (smart contracts). These scripts can either be run from a host machine or from the hyperledger-cli container, which is a container initialized with all the scripts available for execution having the purpose of administration of the network. In the event of the handover of the organization nodes to their respective owners, they will need to follow a procedure to initialize their MSP and PEM files in order to gain full ownership of the nodes, and to communicate their hostnames and public keys to the network admin in order to join the network as participants.

4. UPDATES ON AVAILABLE BRICKS

4.1 User Modelling

4.1.1 Overview

The user modelling module is in charge of creating, maintaining and building a profile of the users of the CPN Reader's App. In addition, it is also in charge of analyzing the articles collected by the Cute4LE module by performing a "semantic enrichment" of the text collected; the multilingual text is extracted in order to find relevant entities (such as person names, organizations names, locations etc) and extracting keywords.

4.1.2 Role

This module constitutes one of the most important elaboration points in the CPN processing pipeline as it lays down the basis for the recommendation module to work. The recommendation process can be seen as a matching process between a list of users and a catalogue of items. Users and items can be described by vectors of features. The module is responsible for creating, maintaining and continuously updating such vectors of features associated to users and news items.

In this first release, the features used to represent Users are the following:

➔ Informational features: name, email contact

➔ Socio-demographic features: age range

➔ Behavioral features: news reading history, like/dislike history

➔ Topics of Interests: the ranked of list of topics for which the user has implicitly expressed interest for over the time

These features constitute the "User profile". The user profile is built partly on the information provided explicitly by the user himself and partly on the information provided by the user in an implicit way (e.g. long time spent reading an article is considered an implicit manifestation of interest for that article and the related topics)

The features used to represent "News Items" are the following:

- Basic article metadata: URL, date, tags (if provided by the original source)
- Automatically extracted metadata: named entities (persons, locations, organizations cited in the text), unsupervised list of keywords, relations between entities in the text.

The module is able to extract metadata from three different source languages, namely English, Dutch and Greek.

4.1.3 Internal Architecture

The module relies on the following components:

- Components for retrieving messages from a broker:

  They are in charge of connecting to the Apache Kafka broker end retrieve different kind of messages: news elaboration messages (new articles to elaborate) and user events.

- NLP processing components:

  Elaboration pipelines able to extract meaningful information from multilingual text. The core technologies actually used are: NLTK, Spacy, Polyglot (python libraries) and Stanford CoreNLP (java library/server).

- Storage components:

  Storing the result of the elaborations and users' personal data: the backend used in the storage module are MongoDB and Apache Solr search server.

- REST API Layer:

  A python Flask layer of services exposing the CRUD operations for user profile management.

4.1.4 API

This module is in charge of maintaining users' data keeping track of his/her interests, history of click and news consumption and demographic data. This data will be mainly used by the recommender engine in order to select the most suitable news in according to a given user profile. The data consumed by this module will be the list of events generated by the users (posted through the broker). The kind of events that the module will be processing are the following:

- "News": when a news item is added to the broker it is retrieved by the module in order to tag, analyze it, extract topics, etc.
- "Users_feedback": every action initiated by a customer that is collected by the platform it is collected, processed and stored in the user profile. Typical events are:

- User clicks
- Users ratings (explicit rates, thumbs up/down, etc)
- User profile update (e.g. change of name, location, age, etc)

Data produced

Schema Version 0.1:

userModellingSchema:

```
{
  "user_id": "String",
  "demo": {
    "gender": "String",
    "age": "Number",
    "name": "String",
    "email": "String"
  },
  "interests": [
    {
      "id": "String",
      "label": "String",
      "score": "Number",
    }
  ],
  "activities": [
    {
      "item_id": "String",
      "event": "String"
    }
  ]
}
```

Module updates: The user modelling profile has been modified in order to include “Locations Of Interest”; this information is given to the recommender in order to privilege articles including entities related to locations of interest for a specific user.

New event types have been added:

- Time spent reading an article
- List of articles skipped by the user (expressing a potential lack of interest for the articles presented)

Finally, the API has been enriched with a method for deleting a particular event (e.g. an article read in the past). This functionality has a two-fold impact: on one side, it gives to the user the complete control of their personal data (including registered events and activity) and it allows influencing the recommender process by deleting, for example, articles not interesting or reading by accident from the activity record.

4.1.5 Testing Scenarios

The module has been tested in isolation and in a full integrated way receiving text from Cute4LE module via the Kafka broker, elaborating and storing into Mongo and SOLR search server. The pipeline is currently active and elaborating all the test documents that are submitted on the broker.

4.1.6 Installation and administration guidelines

The module is deployed as a Docker container into the CPN orchestration platform

4.2 Recommender

4.2.1 Overview

The module is in charge of computing the most suitable news recommendations for CPN users. It has to analyze the users’ profiles and collected news to find the most “interesting” news items to be proposed by the app.

4.2.2 Role

The current state of the art techniques for recommending items are based on two main areas: content based (that relies on good semantic modelling/feature extraction and selection on the items to be recommended) and collaborative filtering techniques (that are essentially domain-independent and take into account network metrics based on emerging similarity graphs of users and items). Our system uses an hybrid approach that uses variable proportions of the mentioned techniques for each user learning (using Machine Learning techniques) from explicit and implicit feedback given by the users themselves: clicks, ratings, sharing, etc. The system is customizable for including content-delivery strategies' optimization: multichannel and date/time optimization (predicting the probability of interests at a given time on a given channel) and includes mechanisms for fostering "serendipitous" discoveries.

The recommender exploits the features extracted from the document enriching modules: relation extraction, user modelling, user feedbacks, topic annotation, uplifting/depressing classifier, semantic uplifting, relation-extractor.

4.2.3 Internal Architecture

The module relies on the following components:

Components for retrieving messages from a broker:

They are in charge of connecting to the kafka broker end retrieve messages that will trigger a new recommendation for a specific user

Recommendation components:

Elaboration pipelines that are used to actually compute the recommendations for all the users: the technique used are content-based recommendations implemented relying on Apache Solr search server and NLP pipelines and "Collaborative filtering techniques" using custom and state of the art libraries such as python Surprise and Implicit.

Storage components:

Storing the result of the recommendation process: the backends used in the storage module are MongoDB and Apache Solr search server.

REST API Layer:

A python Flask layer of services exposing the CRUD operations for recommendation retrieval.

4.2.4 API

Data consumed

The data consumed by this module is the output of DS4Biz-UserModelling module and the output of the news and social media collector modules

Data produced

We will start with a very simple schema for the recommendations to be updated in the next releases.

Model schema:

```
[{
id : String, // Internal Id of the recommendation
user_id : String, // Internal Id of the user
score : Number,  // The relevance score of this recommendation as computed by the recommender engine
date : Date  // When this recommendation was computed
}]
```

Examples

```
[
  {
    "id":"xxxxxx",
    "user_id":"yyyyyyy",
    "item_id":"zzzzzzz",
    "date":"Fri, 11 May 2018 08:40:10 +0000",
    "score":0.8
  },
  {
    "id":"xxxxxx",
    "user_id":"yyyyyyy",
    "item_id":"zzzzzzz",
    "date":"Fri, 11 May 2018 08:40:10 +0000",
    "score":0.5
  }
]
```

Module updates: The recommender has been improved by performing fine-tuning tests on historical data provided by VRT and on data gathered through Pilot 2 execution. The different recommendation techniques have been tested in isolation in order to assess the impact to the overall hybrid technique.

The recommender has also been modified to account for different user event types (registered by the UserModelling module) and adjust the recommendations for each user accordingly.

4.2.5 Testing Scenarios

The module has been tested in isolation and in a full integrated way computing recommendations based on the output of the UserModelling module. The pipeline is currently active and continuously computing recommendations for users as new events occur (e.g. documents are added, user profiles are updated, etc).

4.2.6 Installation and administration guidelines

The Recommender module is composed of several submodules all deployable by using Docker orchestration technologies (e.g., Cattle, Kubernetes, etc.). It is easily deployable using CPN platform's orchestration dashboard (based on Rancher) and a Docker Compose file with all the relevant settings has been provided.

4.3 Topic Extractor

4.3.1 Overview

The module is used to extract topics from articles in order to have a concise representation of the content of the article to be exploited by the recommender module. It is able to perform domain independent terminological, taxonomical and ontological extraction from unstructured sources using metrics and strategies based on statistical, linguistics and extra-linguistic features (e.g. text. The text-extraction sub-module is able to extract text from heterogenous documents including PDFs, web pages, ms-office files, xml, etc. It also provides the possibility of extracting text from scanned PDF documents via state-of-the-art OCR technologies (tesseract v4.x5 [5]). The tool can be used to automatically build a domain terminology/ontology from unstructured sources that can be later used for document annotation and retrieval; the terminological candidates are multiword expressions that are filtered through different stages of scoring. In the context of CPN it will be used to constantly enrich a list of emerging topics from news corpora and annotating incoming news according to this topic list.

4.3.2 Role

The module is operating at the backend level of the CPN platform. It is deployed as a microservice but it is not exposed through the API gateway but it is only queried by integration microservices. The structure of the flow is the following: New content is ingested periodically by the platform by querying the publishers API/RSS/feeds The new content is posted on the CPN message broker (Apache Kafka) A worker module is listening for new messages on the broker and as soon as it receives them it calls the Topic Extraction module by passing the content of the article to it (in the form of a JSON message) The Topic Extraction Module performs a language detection in order to apply the most suitable Natural Language Processing techniques to extract terminological candidates (topics) Topics are filtered and finally ranked according to TF/IDF score and the article, enriched with these topics, is saved into its final storage (Apache SOLR) to be retrieved later during the recommendation phase. The module addresses the user requirement UR-UP1 (Interests: What topics is the user interested in?)

[5] https://github.com/tesseract-ocr/tesseract/wiki/4.0-with-LSTM

4.3.3 Internal Architecture

The module is realized in python using Flask and several Natural Language Processing libraries:

- NLTK: it provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

- Treetagger: it is a tool for annotating text with part-of-speech and lemma information. It was developed by Helmut Schmid in the TC project at the Institute for Computational Linguistics of the University of Stuttgart. The TreeTagger has been successfully used to tag German, English, French, Italian, Danish, Dutch, Spanish, Bulgarian, Russian, Portuguese, Galician, Greek, Chinese, Swahili, Slovak, Slovenian, Latin, Estonian, Polish, Romanian, Czech, Coptic and old French texts and is adaptable to other languages if a lexicon and a manually tagged training corpus are available.

- Polyglot: a natural language pipeline that supports massive multilingual applications.

- Spacy: another natural language pipeline that supports massive multilingual applications.

The module is organized in the following way:

- Service layer: this layer receives the user requests and performs the conversion of the parameters, JSON body and validate them. It then transforms such parameters into appropriate python objects and passes them to the business layer for elaboration. After it receives the response from the business layer it converts it back to a JSON response and gives it back to the client.

- Business layer: based on the user request, as converted by the service layer, it examines the text provide and extracts topics in an unsupervised way by using a noun phrase chunking to build syntactically plausible terminological noun phrases, NPs (e.g. compounds "credit card", adjective-NPs "local tourist information office", and prepositional-NPs "board of directors"). One of the most useful sources of information for NP-chunking is part-of-speech tags. This is one of the motivations for performing part-of-speech tagging in our information extraction system. In order to create an NP-chunker, we will first define a chunk grammar, consisting of rules that indicate how sentences should be chunked. In CPN case, we define several grammars by using regular-expressions rules. The module is invoked directly by a worker (listening to messages on the Kafka broker). The results are then stored on the enriched content storage (Apache SOLR).

Module updates: in order to manage additional languages for different partners/pilots/external media companies the NLP processing pipeline has been extended with several linguistic resources (German language embeddings, Spanish, Italian, etc). Adding support for an additional language can now be, in most cases, performed via configuration files.

4.3.4 API

The Topic Extractor module provides a single API to be used only by internal microservices:

Topic Extraction

Method	HTTP Request	Description
POST	/extract	Extract topics from body content Input format: the service is expecting textual data in one of the currently supported languages (English, German, Greek, Dutch, German, Spanish, Italian) enclosed into a json object with single text key: {“text”: “Donald Trump, the president of US,....”}

Module updates: the overall architecture of the module has been revised for processing tasks in additional languages (Spanish, some tasks for German language, French, etc.). The linguistic resources have been centralized in a data-container deployed on the CPN platform; in order to add a new language support, the data-container can be updated at runtime without the need of interrupting the main Topic Extraction service.

4.3.5 Testing Scenarios

The module has been tested in isolation and in a full integrated way receiving text from Cute4LE module via the Kafka broker, elaborating and storing into Mongo and SOLR search server. The pipeline is currently active and elaborating all the test documents that are submitted on the broker.

4.3.6 Installation and administration guidelines

The module is deployed as a Docker container into the CPN orchestration platform

4.4 Recommender A/B Testing

4.4.1 Overview

The module is tightly integrated with the Recommender module and exposes an API that allow every publisher to create recommenders and user groups. It is a “configuration” module and recommenders and groups can be created and modified in any moment by publisher/administrators of the platform. Recommenders can be created by instantiating the ones currently offered by the platform i.e.:

1. Content-based recommendation: based on unsupervised keyword extraction and named entity extraction, semantic uplifting techniques etc.

2. Collaborative filtering techniques: assessing users consumption history similarity
3. Most Popular Recommendations: recommendations based on trending items
4. Random recommendations: to add variety to the recommendations
5. Composite recommender: quota based combinations of the above techniques

User groups are created by specifying a name for the group and then, by calling a specific api, users are added to groups by specifying their user ids.
The AB-Testing interacts with two modules of the CPN platform: the already mentioned Recommender module and the API Gateway that is needed to publicly expose this internal API to publishers.

- Detail the requirements addressed by this module (referring to specifications codes in table 1/ section 2 of the D3.1: Initial Design and APIs of Technology Bricks).

The module covers a special case of the requirement UR-PS 1 (Detailed Analytics: Giving Newsrooms a more detailed feedback on their audience) since it allows the newsrooms to experiment with different recommendation approaches, collect different feedbacks and formulate insights.

4.4.2 Role

The module is used to be able to test different versions of the recommender at the same time on different user groups in order to compare the behavior of different approaches to content personalization. Every publisher is able to create an unlimited number of groups and recommenders and to associate a specific recommender to a group. A/B testing can produce concrete evidence of what actually works in personalization. The module can be used for continuously testing new techniques and approaches in order to optimize conversion rates and gain a better understanding of the customers.

4.4.3 Internal Architecture

The module is realized in python using Flask and PyMongo libraries.

- Service layer: this layer receives the user requests and performs the conversion of the parameters, JSON body and validate them. It then transforms such parameters into appropriate python objects and passes them to the business layer for elaboration. After it receives the response from the business layer it converts it back to a JSON response and gives it back to the client.

- Business layer: based on the user request, as converted by the service layer, it prepares the queries or the data to be written in the database (by invoking Recommender Factories). The implementation of the Recommender Factory is as follows:

The module is organized in the following way:

```
def factory(request):
    if isinstance(request, RandomRecommenderRequest):
        return
```

```
MostRecentRecommender(None,relative=relativedelta(days=request.days),storage=AppConfig
.ITEM_DAO,users=AppConfig.USERS_DAO)

    if isinstance(request, ContentBasedRecommenderRequest):
        return
CBRecommender(None,relative=relativedelta(days=request.days),solr_dao=AppConfig.ITEM
_DAO,user_dao=AppConfig.USERS_DAO)


    if isinstance(request, CollaborativeFilteringRecommenderRequest):
        return
CFRecommender(None,relative=relativedelta(days=request.days),solr_dao=AppConfig.ITEM_
DAO,user_dao=AppConfig.USERS_DAO)

    if isinstance(request, CompositeRecommenderRequest):
        if not request.sorting:
            request.sorting="score"
        return CompositeRecommender(None, quotas=[RecommenderQuota(factory(recc),v) for
(recc,v) in request.quotas],sorting=request.sorting)

    raise Exception("%s not supported"%request)
```

➔ Data layer: it performs the actual queries on the MongoDB storage by means of a DAO class. The interface of the DAO class is described in the following table

```
class ABTestingDAO:

    def add_to_group(self,group,user_id):
        raise NotImplementedError()

    def map_group_recommender(self,group,recommender_id):
        raise NotImplementedError()

    def recommender_for_user(self,user_id):
        raise NotImplementedError()
```

➔ Any dependencies to external components must be highlighted, as well, as well as providing references to the expected technologies and their specifications exploited by this module.

The module is interacting with the Recommender module and uses as a storage engine MongoDB

4.4.4 API

CPN is a multi-tenant platform for content personalization: each tenant (for example each of the partners of the consortium, DW, VRT and DIAS) has full control on the type of recommendations and user groups. This API is not available to the end-users but only to platform tenants administrators (publishers/media companies).

Recommender management API

Method	HTTP Request	Description
POST	/admin/<tenant>/recommenders	Create a new recommender for the tenant.

Method	HTTP Request	Description
GET	/admin/<tenant>/recommenders	List all the recommenders defined by a specific tenant.

Method	HTTP Request	Description
GET	/admin/<tenant>/recommenders/<id>	Retrieve and show a recommender given its id.

Method	HTTP Request	Description
DELETE	/admin/<tenant>/recommenders/<id>	Delete a recommender given its id

Method	HTTP Request	Description
GET	/admin/<tenant>/users/<userid>	Retrieve the recommender used to personalize content of a specific user.

Groups management API

Method	HTTP Request	Description
PUT	/admin/<tenant>/users/<userid>/<group>	Adding a user to a specific user group.

Method	HTTP Request	Description
GET	/admin/<tenant>/groups	List all the groups defined by a specific tenant

Method	HTTP Request	Description
GET	/admin/<tenant>/map/<group>	Retrieve the recommender used to personalize content of a specific group.

Method	HTTP Request	Description
DELETE	/admin/<tenant>/groups/<group>	Delete a user group.

Recommender/groups mapping API

Method	HTTP Request	Description
PUT	/admin/<tenant>/map/<group>/<recommender_id>	Associate a recommender to a specific group

4.4.5 Testing Scenarios

In the following table, we show a typical json request that can be used to create a composite recommender combining three techniques (Content-based, Collaborative Filtering and Random) with 33%/33%/34% quotas.

POST /admin/<tenant>/recommenders

```
{/admin/<tenant>/recommenders
  "__class__": "CompositeRecommenderRequest",
  "quotas": [
    [
      {
        "__class__": "ContentBasedRecommenderRequest",
        "days": -0.041666666666666664
      },
      0.33
    ],
    [
      {
        "__class__": "CollaborativeFilteringRecommenderRequest",
        "days": 0.041666666666666664
      },
      0.34
    ],
    [
      {
        "__class__": "RandomRecommenderRequest",
        "days": 0.041666666666666664
      },
      0.33
    ]
  ],
  "sorting": "date"
}
```

The output of this call will be a string representing a new recommender id (for example id=5cb903a7c72eacb74cc93b42).

If we have defined a group test we can associate this new recommender to the group by executing the following call:

PUT /admin/VRT/map/test/5cb903a7c72eacb74cc93b42

that can be read in this way: the tenant VRT is giving to the group test the recommendations produced by the recommender 5cb903a7c72eacb74cc93b42.

4.4.6 Installation and administration guidelines

The module is deployed as a Docker container into the CPN orchestration platform

4.5 Reader's App

4.5.1 Overview

The third version of the CPN prototype includes updates on the mobile application and the introduction of a smart speaker application. The user feedback derived from Pilot 2, together with the relative comments from the 2nd Review of the project, were discussed and new features were implemented for Pilot 3,

4.5.2 Internal Architecture

Both the Mobile Android application and the Smart Speaker Application are consumers of the CPN API. Any interaction with the recommender and all the other modules of the platform are achieved via this API.

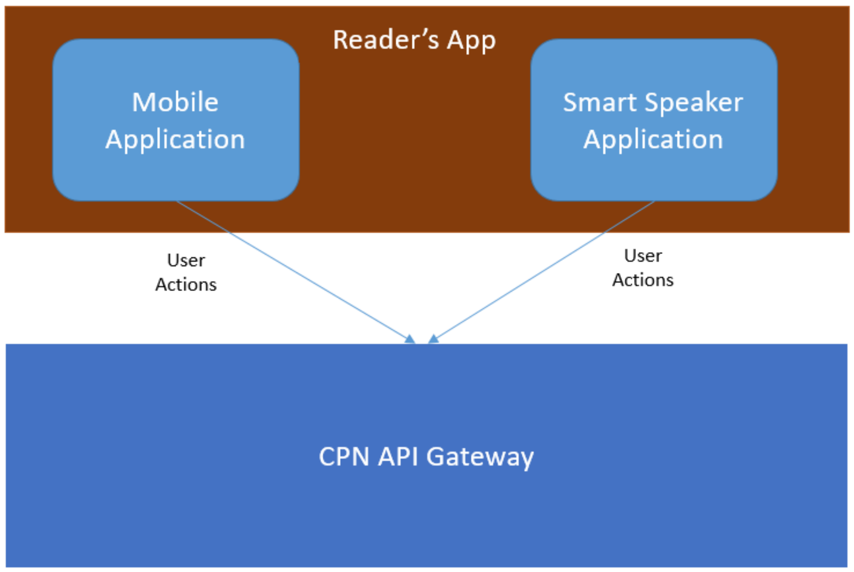


Figure 3: Reader's App consumers

4.5.3 Mobile Application

The mobile application is the main component available for general consumer public. Thus, it is important to achieve an efficient news personalization, making the most out of the Recommender brick and all other bricks that constitute the CPN platform.

The main features of the mobile application that were delivered for Pilot2 are:

Registration/ Login
Within the registration layout, the user enters among others, the preferred media source (non-configurable) and the location of interests. The user can also connect the CPN ID to his/her twitter account. This information, together with the locations of interest are sent via CPN API to the Recommender. Moreover, the user has the control of his/her permissions (location, preferences, time usage). The permissions can be updated any time and the user is informed about this update by means of Personal Data Receipts Brick. For the registration/ login process, the users can also use their Facebook or Google accounts.

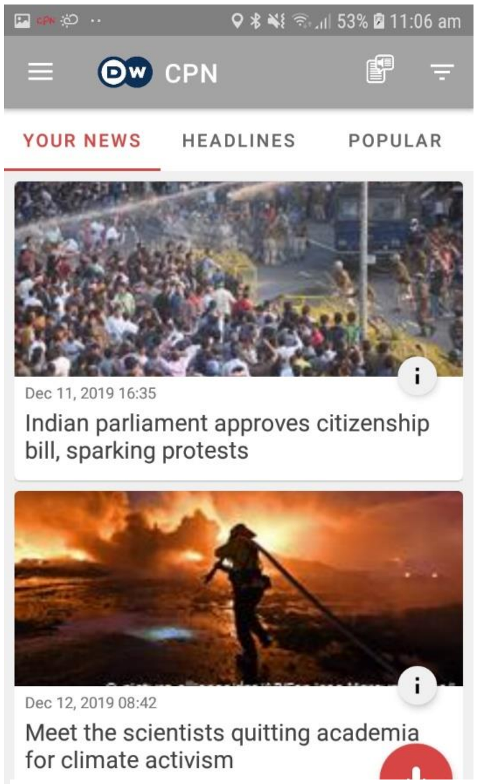

11:06 am
CPN
YOUR NEWS
HEADLINES
POPULAR
Dec 11, 2019 16:35
Indian parliament approves citizenship bill, sparking protests
Dec 12, 2019 08:42
Meet the scientists quitting academia for climate activism

Figure 4: Login Layout

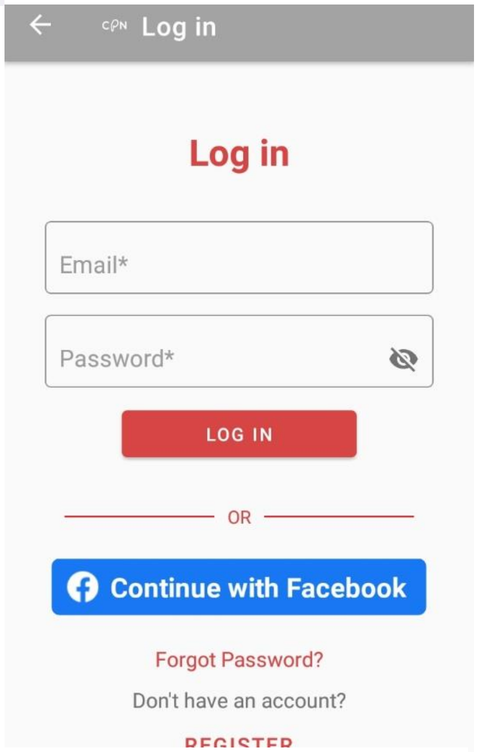


Figure 5: Main Layout

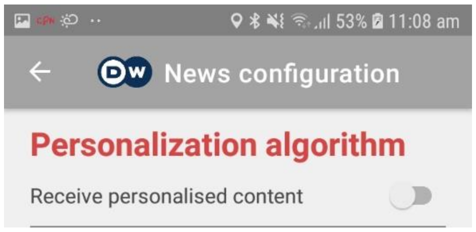
Three Streams in the main layout.
"Your News" stream includes the personalized news information coming from the Recommender.
"Headline" stream includes the most important news, characterized as such by the specific source media.
"Popular" stream includes the most read articles, within the media source available list of articles.

For DIAS (SIgmaLive) media partner, the streams above, are: "Your News", "Latest" and "Popular"

Note that the user can any time deactivate or activate again the personalized information feature

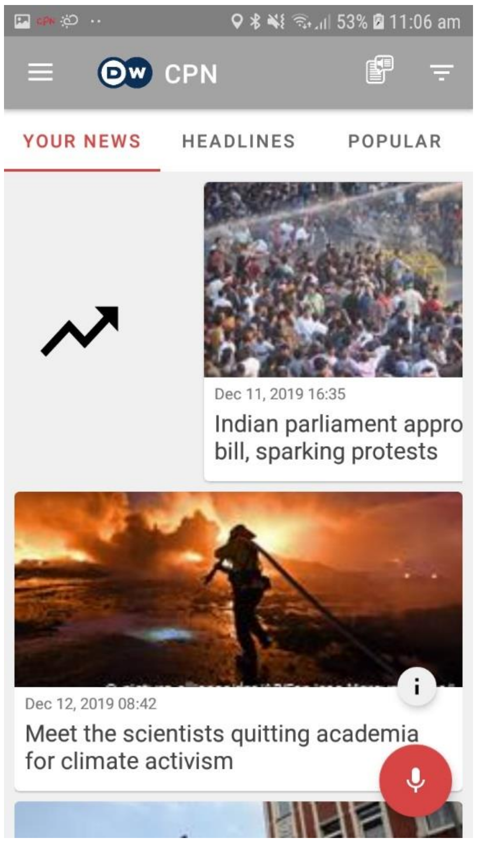
Figure 6: Deactivate or activate again the personalized information feature

Mark an article as interesting or irrelevant.
While using the application, scrolling down within any of the available streams, the user can characterize an article as "interesting" or "irrelevant" by swiping the article right and left, respectively.



Personalization algorithm

Receive personalised content



Dec 11, 2019 16:35
Indian parliament appro
bill, sparking protests

Dec 12, 2019 08:42
Meet the scientists quitting academia for climate activism

Figure 7: Characterize an article as "interesting"

Account settings

The user can at any time connect the CPN account with his/her Twitter profile (to provide more input for the Recommender), can update the permission schema (triggering the Personal Data Receipts mechanism). There is also the option for the user to delete his/her account permanently, with all the data created using the application.

The user activity is tracked, within the application and the following Article related Actions are sent towards the CPN API:

- open an Article,
- read an Article,
- reach the bottom of an Article (full scrolled)
- close an Article,
- interested,
- not interested/ irrelevant

Feedback/ Rating

By means of the feedback or rating feature, both the media and technical partners have the chance to collect a variety of reviews, see what's working and embrace negative reviews.

Additional features added for Pilot 3.
The new version of the mobile application intended to be used for Pilot 3, includes comments from the Pilot 2 and the last official review meeting feedback.

The most important new features included are:

Voice Commands
In Article Details, the user can say:
"read" or "read article" in order to have the title and the content of the article read by the device and its text-to-speech engine.
"interesting" or "mark interesting" to mark the article interesting. After that, the article can be found in the corresponding list of interesting articles.
"irrelevant" or "mark irrelevant" to mark the article irrelevant. After that the article can be found in the corresponding list of irrelevant articles.

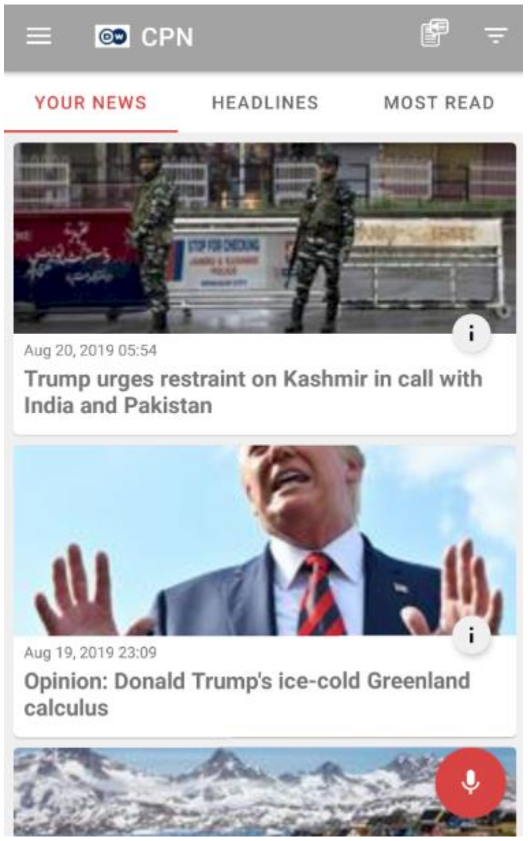

CPN
YOUR NEWS
HEADLINES
MOST READ
Aug 20, 2019 05:54
Trump urges restraint on Kashmir in call with India and Pakistan
Aug 19, 2019 23:09
Opinion: Donald Trump's ice-cold Greenland calculus

Figure 8: Voice Commands

At the Article Lists section, the user can say: “top your news” or “top headlines” or “top most read” to get the first 5 articles of the corresponding stream.

Time Period based Headlines
The user can get headlines from specific time periods (i.e. Latest, Yesterday, Last Week)

Transparency
In each article proposed under the “Your News” tab, the user can know why the specific article is proposed by the Recommender, y means of an “i” on the top right corner.

Facebook login integration
Facebook login is integrated into CPN mobile application. This feature gives the opportunity to use user's Facebook information to enrich their experience and to provide them with tailored content. In order to use this feature the user has to agree on the related policy. The analysis and the evaluation of the content is achieved by u-Hopper SME. The analyzed data is sent from u-Hopper back to the mobile application and forwarded to CPN API.

Text-to-Speech
At the Article Lists section, tapping the related icon, the titles of the articles in the article lists (“Your news”, “Headlines”, “Most read”) are read by the device’s text-to-speech engine while scrolling.
This functionality can be turned on and off by the user.

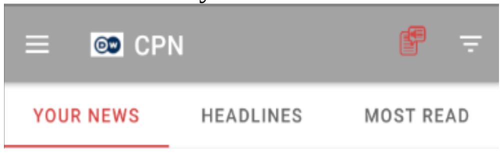
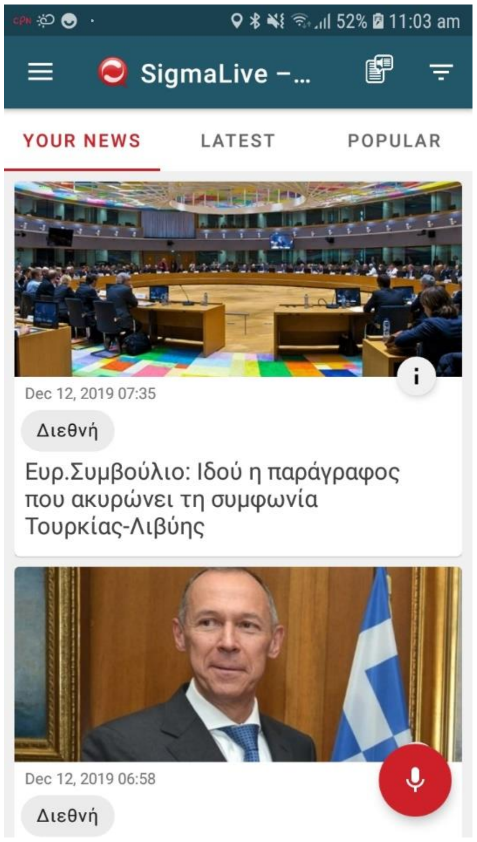


Figure 9: Text to speech button

Location(s) of Interest
The user can add multiple location of interest. If the user has allowed the process of the location data, their current location is considered by the platform as a “location on interest” and vice versa
The information sent towards the CPN API, includes both Name and Coordinates:
User.registrationInfo.interestLocation=
“**Ghent**:lat 51.0543422 lng 3.7174243000000002;
Thessaloniki:lat 40.6400629 lng 22.944419099999998;
CurrentLocation:lat 37.421998333333335 lng -122.08400000000002”



Filtering options
The app let the user to exclude specific categories or tags from the list of the possible recommended articles.

Figure 10: UX adapted for DIAS media partner

Breaking News functionality

Articles characterized as “Breaking” by the media partners, are displayed with the appropriate indication. Breaking news articles are excluded from any filtering settings, since they are of great importance.

Include Surveys within the application.
Instead of communicating with the users via email, surveys are included into the application, using the Qualtrics survey component.

Adapt the UX to be Media source specific.
Since the mobile application is tightly connected to a specific media source, that is not configurable and set during the registration process, the UX environment is adapted to this media source (SigmaLive, DW, VRT). This can add value to the application, taking advantage of the branding of the media partners.

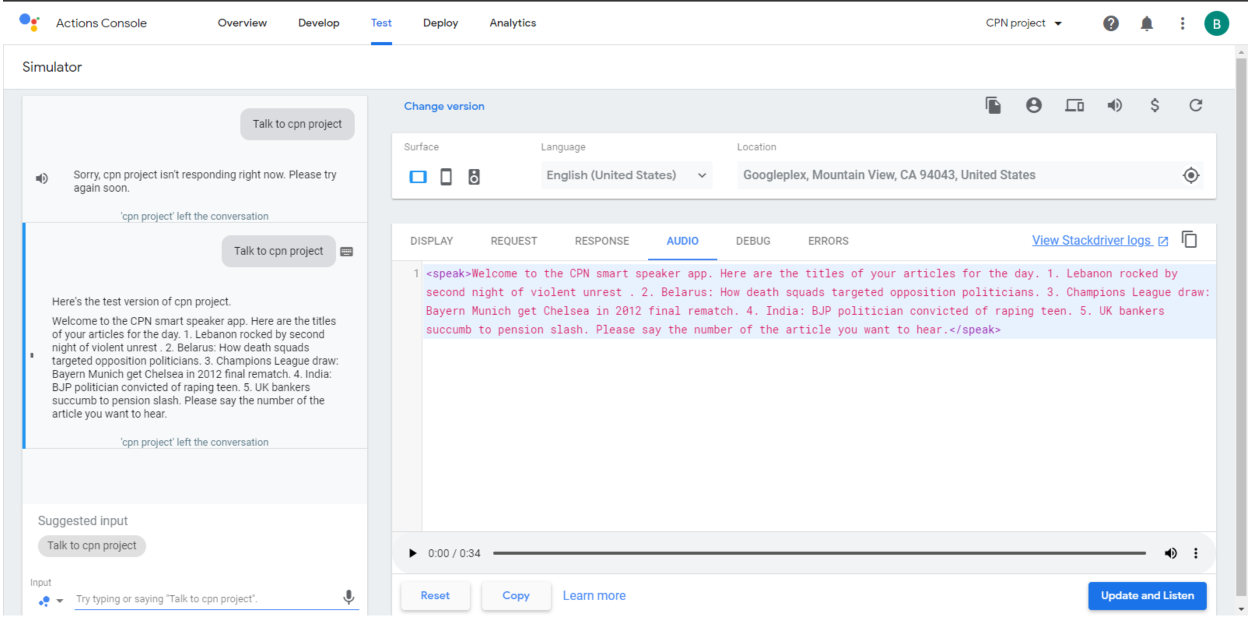
More Features:

- Send towards CPN API the information about the device or the display used.
- Let the backend know how many articles the user has scrolled before opening an article.
- UNDO option.
- Let the user delete specific activities.
- Send “time-spent” action, indicating the time the user spent on a specific article.
- Implement webview for internal links.

4.5.4 Smart Speaker

A Google Assistant application is implemented as an additional Reader’s application.

The smart speaker application will launch, delivering the titles of the recommended articles. These titles will be read by the Google Assistant. The user can ask the Google Assistant to:33



Talk to cpn project

Sorry, cpn project isn't responding right now. Please try again soon.

'cpn project' left the conversation

Talk to cpn project

Here's the test version of cpn project.

Welcome to the CPN smart speaker app. Here are the titles of your articles for the day. 1. Lebanon rocked by second night of violent unrest . 2. Belarus: How death squads targeted opposition politicians. 3. Champions League draw: Bayern Munich get Chelsea in 2012 final rematch. 4. India: BJP politician convicted of raping teen. 5. UK bankers succumb to pension slash. Please say the number of the article you want to hear.

'cpn project' left the conversation

Suggested input: Talk to cpn project

Language: English (United States)

Location: Googleplex, Mountain View, CA 94043, United States

```
<speak>Welcome to the CPN smart speaker app. Here are the titles of your articles for the day. 1. Lebanon rocked by second night of violent unrest . 2. Belarus: How death squads targeted opposition politicians. 3. Champions League draw: Bayern Munich get Chelsea in 2012 final rematch. 4. India: BJP politician convicted of raping teen. 5. UK bankers succumb to pension slash. Please say the number of the article you want to hear.</speak>
```

Figure 11: Smart speaker launched for a logged in user - simulation node

4.5.5 Installation

The mobile application is uploaded on Android Playstore, as a beta version.

The aforementioned link is: https://play.google.com/apps/testing/gr.blockachain.cpn

4.6 Personal Data Receipt

4.6.1 Overview

This module serves to reassure the user that their data is being handled correctly and with their permission. It sends them an email receipt (the Personal Data Receipt) whenever they make changes to the permissions they have granted the system.

4.6.2 Role

This module has an important role for ensuring the user is able to confirm and exercise their rights under GDPR.

4.6.3 Internal Architecture

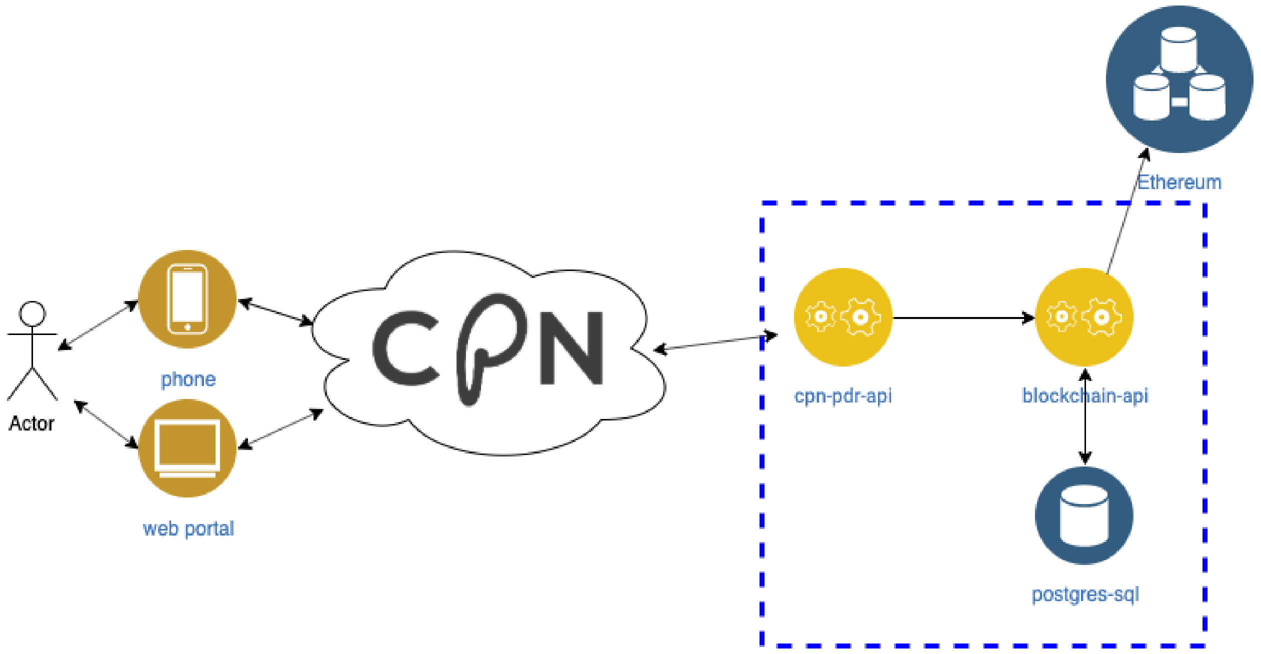


Figure 12: Architecture Schema for Personal Data Receipt module

4.6.4 API

The PDR brick is composed of 2 distincts APIs. the cpn-pdr-api (the public API) and the blockchain-api (internal API used to send the hash to the blockchain).

Both of them expose an http interface; the cpn-pdr-api is in charge of generating the PDR from the user request and to send the PDR to the user via email.

The blockchain-api is called by the cpn-pdr-api to write the hash of the PDR into the blockchain. Continuous checks for any new additions to the ledger are in place in order to synchronize node state. Another loop (transactionEnforcer) checks that all written hashes to the blockchain have reached sufficient block-depth to be considered stable and permanent.

The PDR public API allows to POST a JSON of the following format:

```
{
  "trigger": "PROFILE_UPDATE", // enum { PROFILE_UPDATE | MANUAL_REQUEST | REGISTRATION }, mandatory
  "cpn_user_id": "5b222556f8ac34000a1d1562", // string (base64), mandatory
  "cpn_registered_email": "anthony.garcia@digicatapult.org.uk", // string (email) required
  "user_name": "Anthony Garcia", // string or null (if the user didn't fill the profile section)
  "given_personal_data": [
    // description of what the user entered + justification for the platform to ask for those informations
    { "description": "Email address", "purpose": "To contact the user" },
    {
      "description": "Name",
      "purpose": "Address the user in a more friendly/ personal way"
    },
    {
      "description": "Twitter handle",
      "purpose": "Get further insight on user preferences",
      "shared_with": ["TruthNest"]
    } // The shared key is only present for the data that are shared with 3rd-party
  ],

  "consents": [
    // description of what the user consented to + justification for the platform to collect them
    {
      "description": "Processing of user's location data",
      "purpose": "To recommend content based on user location"
    },
    {
      "description": "Processing of user's time usage data",
      "purpose": "To recommend content based on last user connection"
    },
    {
      "description": "Processing of user's preferences data",
      "purpose": "To generate personalised content"
    }
  ]
}
```

The service will always reply with a valid JSON. The format is either:

- { "error": "<Error message here>" } in case of failure

- The result of the Mailgun sending API

4.6.5 Testing Scenarios

Both services have been written in TDD (Test Driven Development) and have a test coverage close to 100%. The scenarios tested are the different types of user requests (registration to the platform, changes in the consents, manual requests of the PDR) and the different blockchain states (transaction succeeded, failed, pending state…).

4.6.6 Installation and administration guidelines

Both services ship with a Dockerfile, that allow to run them in any Cloud platform supporting Docker or on premises. The services follow the 12factors guidelines and should work with minimum maintenance. Being stateless, they can be restarted automatically in case of crashes. They can be configured using environment variables supplied to the containers. We also provide a Docker Compose file to start the project aimed for development - although the Docker Compose file could be used in production, we don't recommend to do it, and to rather use Docker Swarm or Kubernetes to ensure duplication and availability of the services. The blockchain-api also need a SQL database; we used postgres-sql for development and production.

4.7 Producer's App

4.7.1 Overview

The third version of the Producer's app, as well as including some minor fix respect on the previous versions, extends the Producer's Dashboard UI, provides some new APIs for handling contents and allows to manage the "breaking news".

Concerning the dashboard, three news sections were added:

- Dashboard
- Topics
- Freelancer

The Dashboard section includes an overview on the data collected for a specific news provider.

In particular, some different counts are shown: readings, likes and shares. For each counts the system highlight from which "stream" the action becoming (recommender, most recent, popular), in order to understand the performance of each stream.

For the readings, an extra chart with the timeline is provided and all the data are filterable by date.

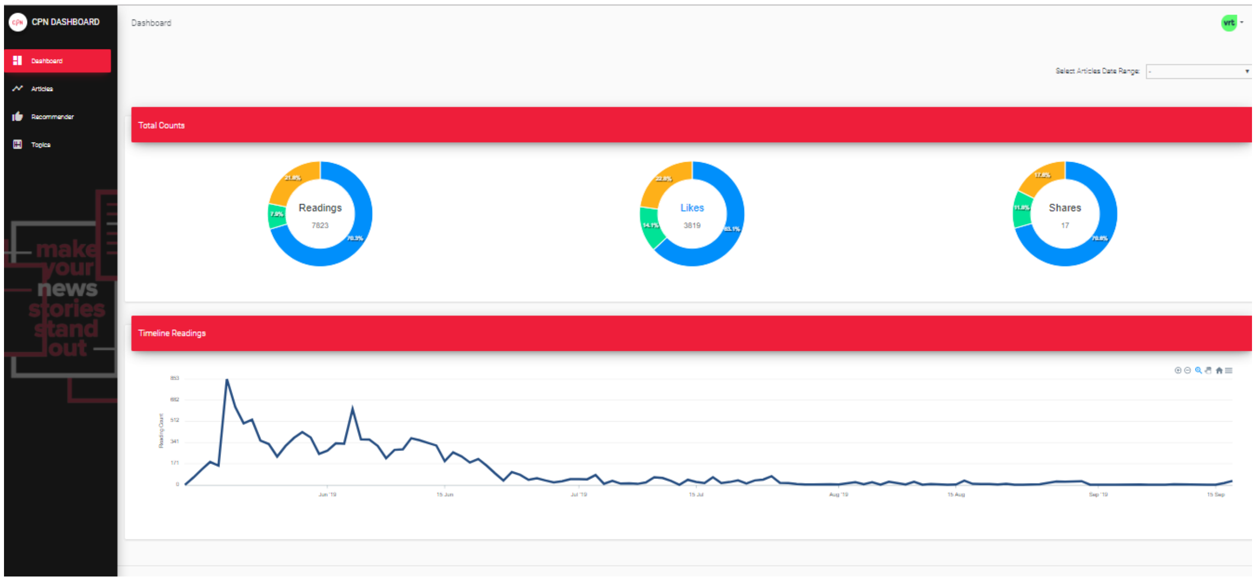


Figure 13: Overview on the data collected

The topic section provides the list of the trending topics for a specific news provider.

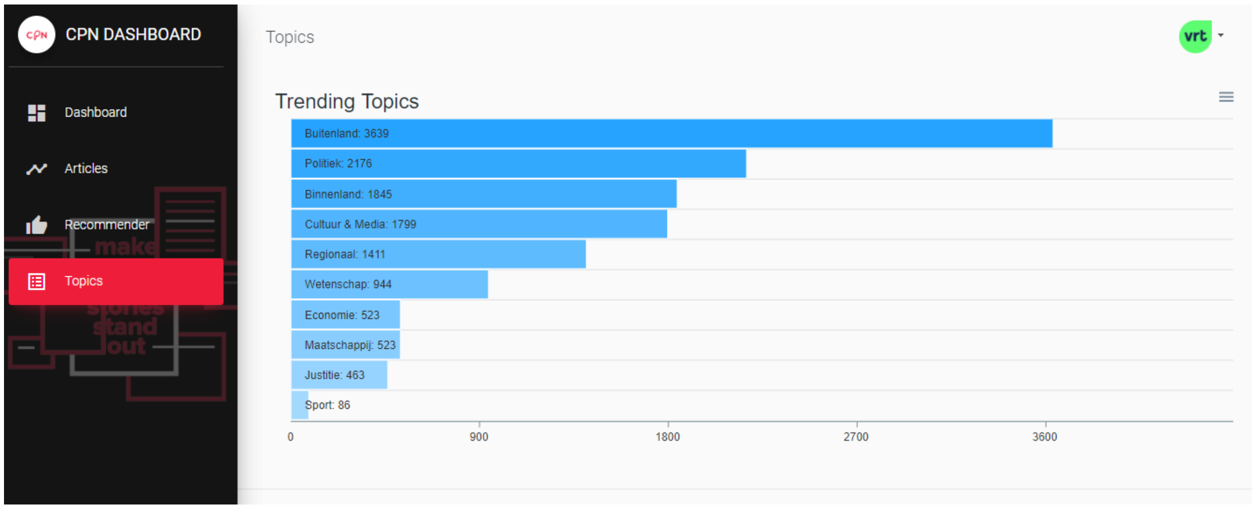
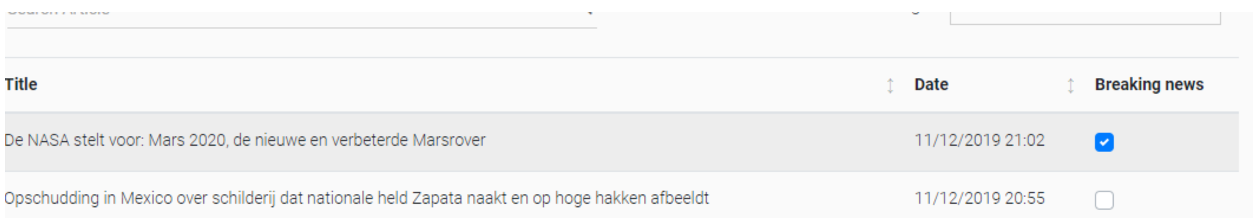


Figure 14: Trending topics for a specific news provider

Even in these new sections, all data can be downloaded in CSV format or extracted in JSON format via REST APIs.

The management of the “breaking news” is now available on article list section. From this section, an editor can now set an article as “breaking news” (see figure 15 below).



Title	Date	Breaking news
De NASA stelt voor: Mars 2020, de nieuwe en verbeterde Marsrover	11/12/2019 21:02	✓
Opschudding in Mexico over schilderij dat nationale held Zapata naakt en op hoge hakken afbeeldt	11/12/2019 20:55	☐

Figure 15: Tag an article as "breaking news"

The producer's app component now includes this information on the article data model and provides a new API for retrieving a list of breaking news. Both these updates were needed in order to manage the "breaking news" within the overall CPN platform.

4.7.2 Role

In the context of the CPN project, the third version of the producer's app addresses the following user requirements:

UR- PS2.1 - The system should allow for an easy integration into the producer's workflow

UR- PS2.2 - The system should provide contract templates to allow freelancers to easily work together and with editors, to define and track the scope of individual contributions and expected revenues

UR-PS2.3 - The system should allow producers contributions are used and distributed to readers

In addition, a new functionality, not initially expected but introduced as improvement of the CPN platform is now supported: handling of "breaking news".

4.7.3 Internal Architecture

No updates on the internal architecture of this module.

4.7.4 API

The following APIs were added on this version of the producer's app:

Method	URI	Input	Output	Description
POST	/v1/admin/article	Article	New Article	Administration service (only news admin can use this API). Create a new article on the CPN system
GET	/v1/breaking	-	List of articles	Returns the list of breaking news for a specific news provider

These APIS are documented with OpenAPI specification[6] (Swagger v2.0) and are testable via CPN API gateway interface.

4.7.5 Testing Scenarios

Compared to the scenario already described for the previous version of the producer's app, now an editor has a complete overview on how users consume its own contents, [s]he can manage the "breaking news" directly from the dashboard UI and s[he] can manage the licensing of the articles provided by freelancers.

All the testing scenarios described for the producer's dashboard, will be evaluated internally to the consortium and a detailed evaluation result will be provided together with the pilot 3 evaluation.

4.7.6 Installation and administration guidelines

No updates on the installation and administration guidelines with respect to the previous version.

5. CONCLUSIONS

This Deliverable reports on the implementation of the technological infrastructure of the third prototype of the CPN platform. This is the last version of the platform components, however, there may be some updates on the currently available components, APIs, and services as a result of the Pilot 3 iteration.

6. REFERENCES

[1] CPN: D1.1 User Requirements Model

[2] CPN: D2.1 CPN Reference Architecture

[3] CPN: D3.1 Initial Design & APIs of Technology Bricks

[4] CPN: D3.3: Technology Bricks V2

[5] CPN: D2.4: Open Virtual Platform V3

[6] CPN: D6.5 2nd Review Periodic Report

[7] Docker Container Platform

[8] Apache Kafka distributed streaming platform

[9] Processing tools or libraries for Natural Language Processing

[10] Apache Solr search server

[11] RESTful webservice

[6] https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md