Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Introducing the Dynamic Nuchwezi Architecture Platform

Joseph Willrich Lutalo*
j.lutalo@cis.mak.ac.ug

May 27, 2020

Abstract

A novel software engineering platform called the Dynamic Nuchwezi Architecture Platform (DNAP) is introduced, specified and its novelties explained. The unique features of this platform are explained and several new concepts and abstractions upon which its implementation, usage, and analysis are hinged also elaborately discussed. The motivations for this new approach to building especially tools used in data engineering are spelled out and the platform is contrasted against other existing technologies of a similar kind. Finally, it is shown what known limitations DNAP suffers, as well as what room for further research and improvement there is in this field.

Keywords: Cross-Platform Development, Citizen Programming, JSON Schema, Data Engineering.

1 Introduction

First conceptualized in 2016, Dynamic Nuchwezi Architecture Platform or simply "DNAP", was originally inspired by a growing demand in the business sector, of an otherwise scarce software skill then - the ability to design and deploy dynamic data collection and analysis solutions accessible to anyone in an organization, on-demand and at low cost. These skills we call data engineering skills, and we found that many businesses already owned some off-shelf or custom-made software that to some degree allowed for data engineering, but also that extension of these tools to new data types - such as data via OCR, Barcodes or GPS or capabilities such as being able to work offline was not so easy, and in some instances wasn't even supported. Further, we found that the combination of data acquisition and data dissemination in the same tool — as might be required in certain contexts, was not readily possible without turning to a

*Master of Science in Data Communications and Software Engineering, Makerere University, College of Computing and Information Science

© 2020 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

full-fledged general purpose programming language. We found these problems to be surmountable with new approaches, new solutions to an otherwise old problem in the data engineering and business intelligence domains and that is how DNAP came to be.

We also realized that if the method of building new software or extending existing software with new functionality were made so simple and readily accessible to those without expert knowledge of general purpose programming languages, then the unavoidable increase in demand for data engineering skills in today's increasingly automated society would readily be satisfiable. This would boost and make very affordable the automation of many business and personal processes than is currently possible.

The key objective of our research then, was to design and implement a new approach to software construction and extension that would readily be applicable to the identified gap in data engineering and business intelligence. This meant that the approach to be developed would have to be accessible to those with low or no experience using standard solutions such as general purpose programming languages.

More specifically, the research focused on the implementation of a software engineering platform spanning the web and mobile, and which platform would allow the design and implementation of simple programs for data engineering especially, without requiring that users know or learn to write code by hand. The idea was to ensure that this platform would make possible the following at minimum:

- Building basic data acquisition apps in the form of survey tools
- Building information dissemination apps with media-capabilities
- Building live documents with both input and output capabilities
- Making the extension of existing mobile and web software easy using embeddable components
- Making the construction of and access to data APIs from such tools direct and configuration-free

Work on the solution started in 2016 at the Nuchwezi ICT Research Lab, as part of a project originally called "Project Okot" or simply "Project O". This work went through several engineering and architectural upgrades since then, and the result is what is now called the Dynamic Nuchwezi Architecture Platform.

In the rest of this paper, we illustrate and explain what exactly this newly realized approach to software development and extension is, and how it gets to fill the identified gap in data engineering first of all. We shall particularly be concerned with understanding the architecture of DNAP as used to realize this new approach, and we shall see how it can be applied in practice. Lastly, we shall explore what limitations DNAP faces, and then call-out what remains to be done with this new approach to software and data engineering.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

2 Existing Solutions of a Similar Kind

First of all, DNAP spans the following categories of technologies;

1. Visual Data Tool Development
2. Data Tool Scripting
3. Offline Data Storage
4. Data Service Discovery
5. Data Service APIs
6. Data Service Composition
7. Dynamic Exploratory Data Analytics
8. Mobile and Web Embeddables

In this paper though, we don't try to expound on each of the above. Instead, we call out how other similar technologies compare to DNAP in terms of completeness based on the above categories, and then only go in-depth for a few of these, with the aim of focusing on what makes DNAP ideal for data and software engineering today.

There are several proprietary and open-source technology platforms mentioned in the literature and community reviews, that are similar to DNAP in the context of data engineering platforms, but which either target a different kind of user or which only span one or only a few of the categories listed above. This is important, as each category calls out a capability, and so, tells us about the applicability — in the case of those platforms supporting it, or the limitations - for those which lack it, of a technology platform in the domain of data engineering as we present them hereafter.

We particularly chose the following technology platforms as a fair sample of the state-of-the-art in data engineering solutions that support a Data Tool Development capability at minimum.

- DataScope [Dat19] — is proprietary, features a web-based form builder, mobile-able forms and data analytics
- Kinto Formbuilder [Kin19] — is open-source, features a web-based form builder, web-based forms and data storage, but no mobile-capabilities
- Zoho Survey [Zoh19] – is proprietary, features a web-based form builder, mobile-able forms and data analytics
- Google Forms [Goo19] — is proprietary, features a web-based form builder, web-based forms and spreadsheet integration for storage and analysis

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Table 1: Comparison of Data Acquisition Platforms

	DataScope	Kinto Formbuilder	Zoho Survey	Google Forms	DNAP
Visual Data Tool Development?	1	1	1	1	1
Offline Data Storage	0	0	1	0	1
Data Service discovery and access?	0	0	0	0	1
Data Service Composition?	0	0	0	0	1
Data Tool Scripting?	0	0	0	1	1
Data Service APIs?	0	0	0	1	1
Dynamic Exploratory Data Analytics?	1	0	1	1	1
Mobile and Web Embeddables?	0	0	0	1	1
SCORE	**2**	**1**	**3**	**5**	**8**

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

The above technologies and the reference list of capabilities, looked at side-by-side with DNAP, exhibit the patterns shown in **Table 1**.

From **Table 1**, we can see that the closest contender to DNAP is Google Forms. However, it lags behind in some respects; for instance, it does not support offline use, doesn't allow for auto-discovery of tools or datasets generated over the platform, and doesn't make use data APIs readily accessible when authoring tools or chaining them — all of which DNAP sets out to address.

On the other hand, we find that solutions such as Kinto's Form builder [Kin19] offer only the most minimal set of features - basically a means to craft a web-based data collection tool, the data collection interface and not much else. DNAP on the other hand set out to make creation of both native mobile and web interfaces for data collection possible from a single app definition, and in the following sections, we shall especially see how this is realized, as well as what the actual novelties are that we introduced.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

3 The Architecture

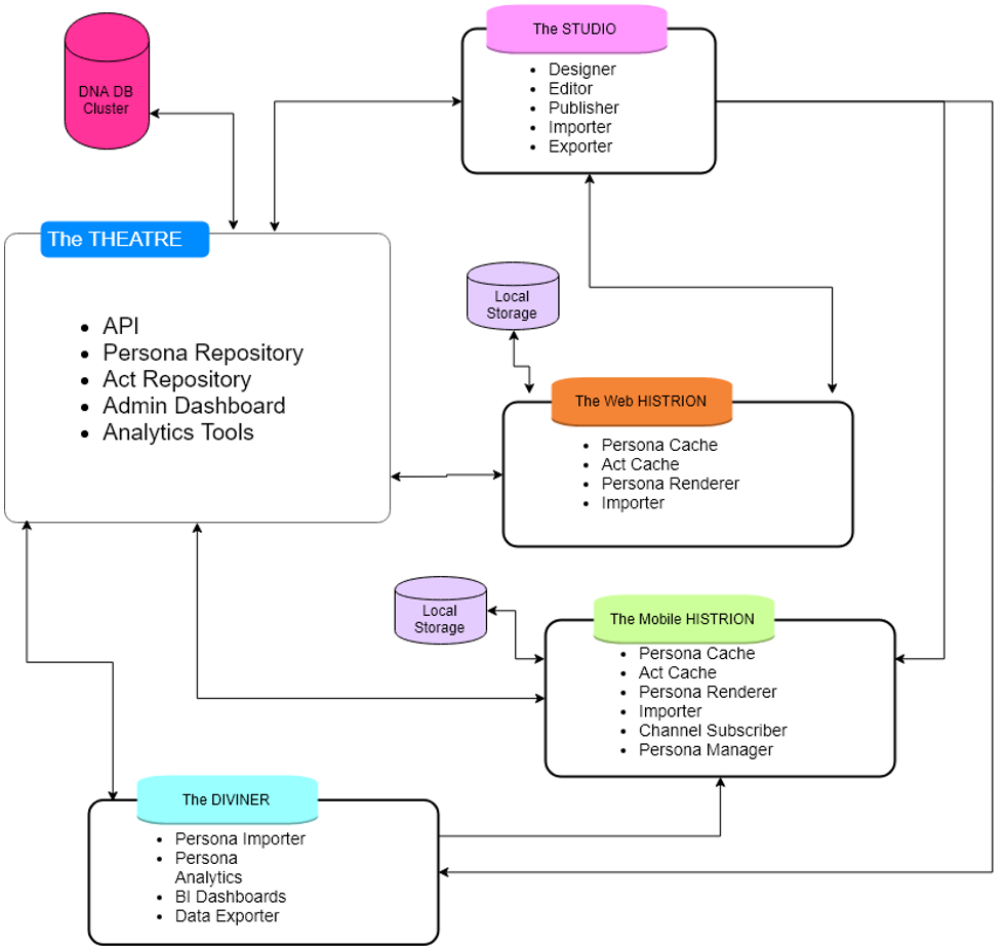

Figure 1: Dynamic Nuchwezi Architecture Platform

In **Figure 1** are highlighted the key components of the Dynamic Nuchwezi Architecture Platform from an architectural perspective, and **Figure 2** attempts to show how each component functions using a mindmap. However, to fully understand DNAP's structure, it is important to first gain a familiarity with and an understanding of the terminology used to describe its components as well as gaining an understanding of each component's roles and their relation to the whole.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

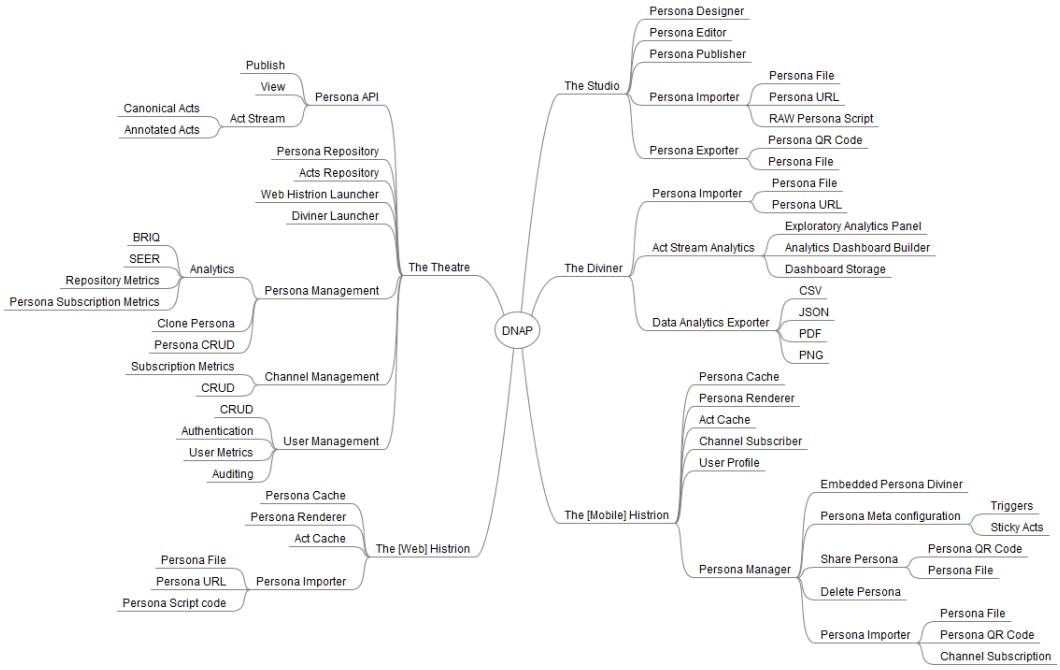

Figure 2: DNAP Overview

The following sections introduce each key concept, and explain its role and relevance within DNAP, in detail.

3.1 Persona

DNAP is first of all a data acquisition, storage and analysis platform. However, in the context of DNAP, "data" refers to a kind of structured data we call "acts", and whose structure is defined by specific schemas. These schemas, specified using JavaScript Object Notation — JSON [Bra17] and adopting some of the ideas of JSON Schema [AW19] — which is a standard for annotating and validating JSON data, enable a mechanism for defining or specifying a new kind of application that we refer to as a persona applet.

> **Definition 1: Persona:** *A data structure, holding the definition, constraints and meta-data of a data application on the Dynamic Nuchwezi Architecture Platform.*

> **Definition 2: Persona Applet:** *A data-driven application running natively on mobile or web, and whose behavior and looks are derived from its associated persona.*

A typical persona,"persona script", "persona source code" or "persona file" has the following general structure:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Listing 1: A Persona Structure

```
{
        "app": {
                "name": "<APP_NAME>",
                "color": "<HEX_ENCODED_COLOR>",
                "theatre_address": "<URI>",
                "channel": "<PERSONA_CHANNEL>",
                "transport_mode": "<POST|GET|SMS|EMAIL>",
                "description": "<APP_DESCRIPTION>",
                "brand_image": "<APP_BRAND_IMAGE_URL>",
                "uuid": "<PERSONA_UUID>"
        },
        "fields": [{
                        "field_type": "<FIELD_TYPE>",
                        "cid": "<CID>",
                        "pattern": "<REGEX>",
                        "required": <BOOLEAN>,
                        "label": "<FIELD_LABEL>",
                        "field_options": {
                                "description": "<FIELD_DESCRIPTION>"
                        },
                        "meta": "<FIELD_META_PARAMETERS>"
                },...
        ]
}
```

For demonstration purposes, in the rest of this paper, we shall use as reference, a persona designed to crowd-source local weather updates and especially occurrences of rain. This persona we have called "NJURA", and its full definition can be studied in **Appendix A**.

3.2 The Studio

Having seen what a "persona" is in the previous section, we can then formally define the "studio" as:

> **Definition 3: Studio:** *An online integrated development environment (IDE), for the design, editing and publication of personas.*

We shall also need the following key concept:

> **Definition 4: Channel:** *A namespace identified by a non-empty alpha-numeric string, to which personas can be published at design time, via the studio.*

Knowing that personas can describe classical data collection tools such as surveys, polls, questionnaires and the like — also known as "forms" or "data tools", it makes sense to contrast the studio with what other technologies do similar work.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Ignoring the rest of the DNAP and only focusing on this component, the closest technology in terms of intent, is Google Forms [MB13], which one researcher described as a Web application that allows users to create surveys and polls that can be distributed to authorized users [DIP15]. However, unlike Google Forms, the studio does more than just allow the authoring of data acquisition tools; persona applets can do more than just collect data; you shall see by inspecting what field types are supported (see **Table 2**), that the studio also enables the design of information dissemination tools such as basic media players (audio and video), e-zines (mixed media, including webpage embeds) and the like.

The process of designing and publishing a tool though, remains somewhat similar; the Integrated Development Environment, IDE, is web-based, and is typically best used from a large view-port such as a laptop, desktop or tablet, though, in practice, we have managed to author some tools via the reference implementation studio, on a constrained smartphone's web-browser without trouble. Also note that the core feature of this IDE is to allow visual as opposed to textual coding or programming.

As of this moment, the reference studio implementation's interface looks like this:

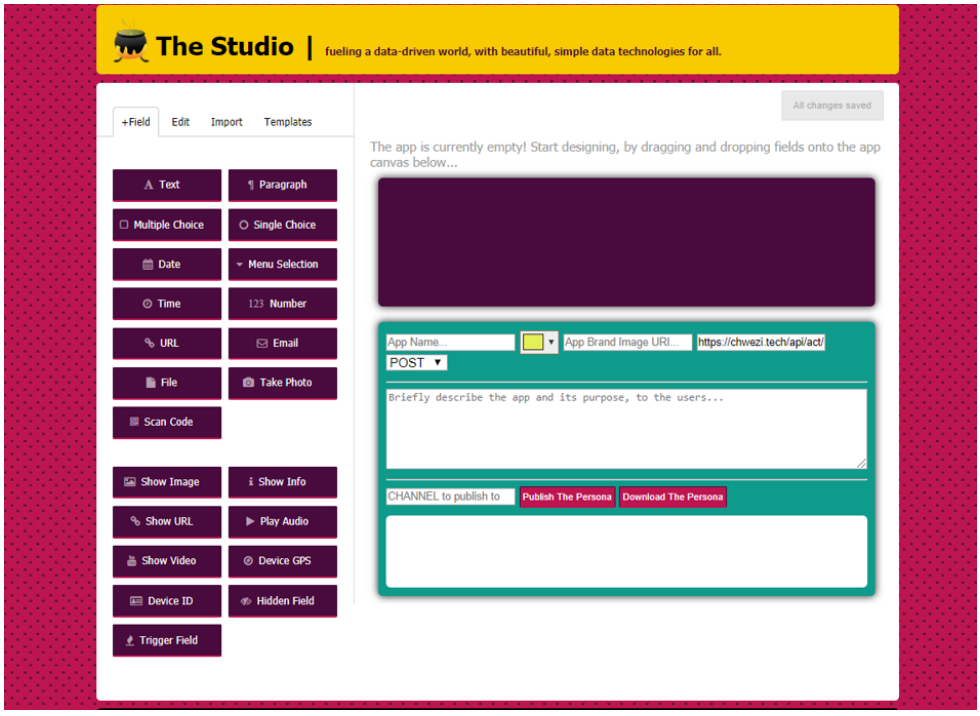

Figure 3: The DNAP Studio — an online IDE for personas

Using the studio interface, a typical approach to developing a persona might proceed like this:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

1. Given a problem for which you wish to collect or share information, determine the set of fields that need to be presented to the user, for purposes of collecting or sharing information as part of the desired solution.

2. Then, for each field thus chosen, determine the kind of field-type best suited for it on the target app. **Table 2**, though not exhaustive, summarizes the available field type options on the reference studio — and consequently, what DNAP natively supports.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Field Type	Parameters	Potential Applications
Text	• Label — the text title above the actual input field on the final app interface. • Description (rendered as a smaller label below the field, on the final app interface) • Validation Pattern — a regular expression to be used to validate the structure of user-supplied values on the final app • Required — a boolean flag indicating whether the field is mandatory or not	• Capture any data that can be specified as a few words (no more than a sentence recommended) using letters, numbers and or symbols.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Field Type	Parameters	Potential Applications
Paragraph	• same as for Field Type "Text"	• Good for cases where text is captured that might be a sentence or more, and where the input could be expressed using paragraphs. • Notes, poetry, messages, etc.
Multiple Choice	• same as for Field Type "Text" • Options — one or more parameters that specify the alternative values that a user can pick from.	• Scenarios where the data entrant is expected to answer with one or more options from a provided list • E.g. favorite foods, games played, subjects studied, etc

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Field Type	Parameters	Potential Applications
Single Choice	• same as for field-type "Multiple Choice"	• Scenarios where the data entrant is expected to answer with ONLY one of the options from a provided list. • E.g. Ones gender, ones ethnic group, marital status, etc.
Menu Selection	• same as for field-type "Multiple Choice" • In the parameter field "Meta", it is possible to point to a URL from which values used in populating the field are to be fetched (at run-time). These values are expected to be encoded as a flat JSON array of strings.	• same as "Single Choice", except this field type is rendered as a drop-down menu on the final app

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Field Type	Parameters	Potential Applications
Date	• same as for field-type "Text"	• Used in specifying dates via a calendar or date-picker widget • E.g Date of Birth, Date today, etc.
Time	• same as for field-type "Text"	• Used in specifying time via a clock or time-picker widget • E.g. Planting time, Appointment time, Now, etc.
Number	• same as for field-type "Text" • Minimum — lowest accepted value • Maximum — highest accepted value	• Useful when capturing numeric values — integers or decimals supported. • E.g. Weight, height, speed, count, position, etc.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Field Type	Parameters	Potential Applications
URL	• same as for field-type "Text"	• Useful when capturing links to websites, documents or just web addresses of any kind.
Email	• same as for field-type "Text"	• Useful in capturing valid email addresses.
File	• same as for field-type "Text" • Has a "Mime Type" parameter to help constrain input files to only specified types	• Useful in attaching and submitting files alongside other data.
Take Photo	• same as for field-type "Text"	• Useful in capturing an Image using the camera on the device where the histrion is run.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Field Type	Parameters	Potential Applications
Scan Code	• same as for field-type "Text"	• Allows scanning barcodes and qrcodes at runtime, as an input method.
Device GPS	• Has only one field, "Field Name" used to name the parameter in which GPS coordinates in the form *Lat,Lng* are to be stored when the persona runs.	• Useful in geo-tagging data captured on the move, especially via mobile devices. • Holds the most recent GPS reading registered by the device while the data entry session runs.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Field Type	Parameters	Potential Applications
Device ID	• Has only one field, "Field Name" used to name the parameter in which a device GUID is to be stored when the persona runs.	• Useful in uniquely identifying data posted from a specific device, and for evaluating how many distinct devices engaged in a given data collection exercise • Also helps with associating devices with users, thus improving in auditing — e.g. where users post data using similar usernames.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Field Type	Parameters	Potential Applications
Show Image	• "Label" field specifies title of the image • "Paste URL to Image" to specify a direct URL to an image file	• In designing the UI of persona applets: by linking to and thus displaying online images on the target persona — E.g. one can display the logo of a company at the top of their survey or add an image to an e-magazine persona. • Also helps with building info-apps such as e-magazines, periodicals or dynamic manuals.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Field Type	Parameters	Potential Applications
Play Audio	• same as for field-type "Show Image"	• Building Media apps: by enabling audio playback on the target persona — E.g. one could publish a weekly podcast, by linking to the podcast audio file, and then could use DNAP to collect listener's feedback on the audio show via extra fields such as comments, rating and the like.
(and many other types...)		

Table 2: Field Type Specification on DNAP

3. Drag-and-drop the desired field type onto the canvas, at the position where you wish the target field to be rendered on the final app. Note that only single-column views are currently supported.

4. Go ahead and configure the field; specifying the label, description and other parameters as required (see **Table 2**)

5. Continue doing the above for all fields needed on the app — it is acceptable to add the same field type multiple times when designing your persona, as the studio assigns unique ids ("CIDs") to each field.

6. Once the app's fields are fully specified, you can then specify the few app-specific parameters:

    - App Name — name of the app (or rather, name of the persona)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

- App Description – one or more sentences explaining the purpose and usage guidelines for the persona
- Transport Mode — how the data from the persona should be submitted to the backend (options include HTTP POST, HTTP GET, SMS and EMAIL)
  - Note that SMS and EMAIL only make sense where the persona is expected to be used on a mobile histrion (refer to **Definition 5**) — such that captured data is encoded as text and submitted to either a target phone number or email address (see *Theatre Address* below)

7. Theatre Address: for each designed persona that is expected to be used for data capturing, it is required to specify where the data is to be submitted. By default, The studio will pre-fill the "Theatre Address" parameter with a default destination — typically a URL to the reference DNAP theatre implementation — *chwezi.tech*, but this value can be overridden as needed, at design time.

   - **Note**: when the "Transport Mode" is SMS, the Theatre Address is expected to be a fully-qualified phone number — including country code that is, and where it is EMAIL, this parameter expects a valid email address.

8. App Color: The studio also allows the persona designer to specify a base theme color, which would then be used to generate a multi-color palette with which the final persona applet and its fields are styled during their rendering so as to readily distinguish the persona applet from others and to help communicate the applet's intended purpose or brand for example.

9. Persona Channel: The studio also offers a means to publish one or more personas via a "channel" (see **Definition 4**). This feature is intended to help logically group related personas on the DNAP persona directory, and also to make it easy for end users to subscribe to one or more channels, so as to have any personas published under those channels get automatically fetched and installed on their devices. The channel can be a simple string such as "ACCOUNTS" – so for example in an organization using DNAP, tools meant for M&E in the accounts department are separately discoverable and deployable without confusing them with those for the sales department, which can alternately have theirs published under a conveniently named "SALES" channel.

10. Once all necessary app parameters are specified, the persona can then be published via the studio, by clicking on the "Publish the Persona" button. Note that persona publication does the following:

    (a) The persona specification (the persona script defining it that is), is lexically verified on the studio

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

(b) Once validated, the persona specification is then submitted to the theatre for storage.

(c) On the theatre, this persona is assigned an App UUID that is expected to be globally unique.

(d) If the persona specified a channel, that channel will be created if it does not already exist.

(e) On the theatre, the published persona gets added to the list of available, active personas, such that DNAP clients (histrions, diviners, etc.) can start to reference the persona via its UUID for various purposes.

(f) After processing the persona, and submitting the results back to the studio, a special QR Code — a "Persona QR Code", is generated, that is then displayed on the studio where the persona was published from. This code can be downloaded, printed or used as is, to help the app developer or users fetch and install the target persona on a compatible mobile device running a compatible histrion or other DNAP-compatible app, without the need to explicitly quote or know the associated persona UUID or its online location.

11. Alternatively, instead of publication, once the persona is designed, it can merely be downloaded from the studio, as a "persona file" — signature **.persona*. This is done by clicking on the "Download the Persona" button on the studio. This persona file can then be used to distribute the persona online or offline, and can later be installed in histrions via the "Install from File" feature for example.

3.3 The Histrion

Once a persona has been developed — via the studio, or by hand, a means to turn it into an app that someone can then interact with is necessary. The "histrion", which is available both as a mobile and web app as of this writing, is what plays this role in the DNAP architecture. It is the "front-end" of DNAP, and precisely specified, is:

> **Definition 5: Histrion:** *A general purpose app that can parse and interpret persona source code from which it then renders persona applets.*

In a sense, a histrion is akin to a web-browser, only that instead of HTML, CSS and JavaScript, the histrion processes a special JSON encoded document called a persona, and instead of mere web pages, it outputs persona applets (see **Definition 2**).

As its name implies, this technology behaves more like an actor in a theatre; capable of assuming new roles and looks on-demand, based on the currently active script — in our case, the "script" being the "persona script" introduced

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

earlier on. Looked at this way, it could be argued that most (if not all) existing web-browsers are a kind of histrion — in which case a persona script is analogous to a document written in markup language, but that's where the similarity stops.

As of this moment, there exist two reference implementations of a histrion — one for Android mobile devices [Lut19] and one for the web [Lut20b]. Both of these can take a persona and output a compatible persona applet that a user can interact with, though currently, the mobile implementation of the histrion surpasses the web version in terms of feature completeness — for example, the web histrion does not yet support the "Show Video", "Play Audio" or "Device GPS" field types, though, it supports enough of the other basic types to serve its purpose as a platform for the implementation of surveys, polls and basic data acquisition and data dissemination tools.

To get an appreciation of what the histrion does, here is a simplified architecture of the Reference Implementation ("RI") Histrion.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

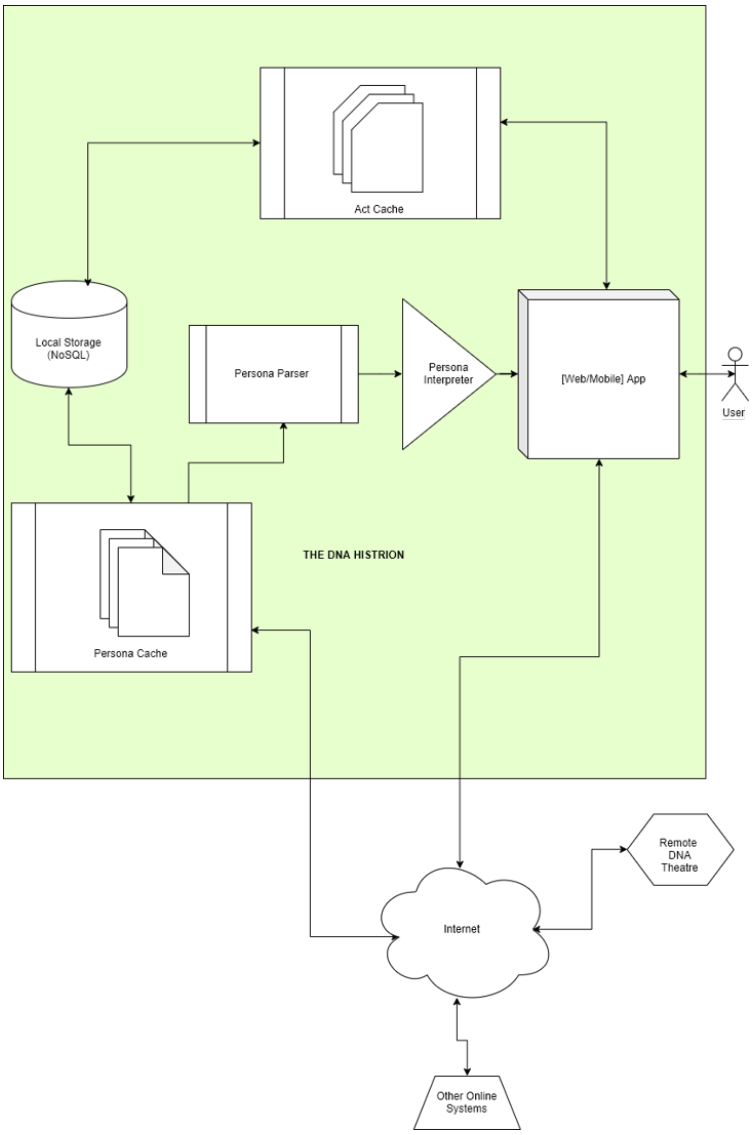

Figure 4: RI Histrion Architecture

The key highlights of a histrion include:

1. It is a standalone app, but whose purpose is to load and run other apps — the personas.

2. Personas published via a compatible theatre (such as *chwezi.tech*) can be discovered and can be locally cached via user-configured subscriptions to "channels" (see **Definition 4**).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

3. It can load personas from a local (persona file) or remote/online source (persona QR Code or channel subscription)

4. It renders a persona as a native app depending on where the histrion is running — mobile or web

5. Its persona cache allows a user to run or interact with a persona originally fetched online even if they go offline

6. It can allow offline data collection — data records thus collected also known as "acts" (see **Definition 6**)

7. Via an embedded diviner (see **Definition 10**), it allows a user to browse or analyze submitted acts for any loaded personas

In addition to these core features, the current mobile implementation of the histrion [Lut19] allows for basic user identification and tagging — at installation time and or via the default histrion settings; the user specifies a name and contact, which information is automatically included in each and every act, so that later, together with the use of the "Device ID" field type on a loaded persona, it becomes possible to unambiguously distinguish and attribute data submissions from multiple devices to their respective owners/authors and enhance traceability of origin or simplify auditing where users, such as data enumerators are to be paid say based on their actual submissions. This mechanism allows for "non-repudiation", a very important factor in data and information security.

To see what a histrion loaded with a persona looks like, refer to **Appendix B** and **Appendix C** — for an example of how the persona specified in **Appendix A** looks like on the web and mobile histrions respectively. Otherwise, you can see a blank, "vanilla histrion" by installing the reference mobile histrion [Lut19] or visiting the web histrion [Lut20b].

3.4 Acts

As we have already seen, the key purpose of DNAP is to collect structured data via compatible histrions. In this regard, when a persona is used to capture some set of pre-defined data fields via a histrion, the resulting data record, which is encoded using JSON, is what is called an "act".

> **Definition 6: Act:** *A data record whose structure is a flat JSON dictionary with keys corresponding to fields on a persona, and the values holding data recorded against each field during a data collection session, typically via a histrion.*

Thus we know that given an act, we expect there to exist a persona specifying its data semantics. In general, an act has the following structure:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Listing 2: Act JSON Structure

```
{
    "<FIELD_ID>":"<FIELD_VALUE>",
    "<FIELD_ID>":"<FIELD_VALUE>",
    "<FIELD_ID>":"<FIELD_VALUE>",
     ...
    "OBSERVER":"<USER_NAME>",
    "OBSERVER_ID":"<USER_ID_OR_CONTACT>",
    "CACHE_TIMESTAMP":"<YYYY-MM-DDTHH:mmZ>",
    "UUID":"<PERSONA_UUID>"
}
```

The extra fields not labeled <FIELD_ID> are "meta fields", that help to classify or track certain aspects of the data collection session that produced the data held in the <FIELD_VALUE>entries.

For example, given a persona such as "NJURA" (refer to **Appendix A**), the following is a valid act:

Listing 3: Example Canonical Act

```
{
    "c2":"1 Nuchwezi Road.",
    "c10":"NORTH",
    "c22":"8",
    "c18":"YES",
    "c6":"0.0903518,32.5429456",
    "OBSERVER":"P.",
    "OBSERVER_ID":"0700100000",
    "CACHE_TIMESTAMP":"2019-10-10T00:25Z",
    "UUID":"c6e47f79-afd9-4ebc-8753-3da5aa5f4cd7"
}
```

You will note that this act instance is not readily decipherable without proper context or extra information — say, access to the specification of the persona applet or the persona, via which this act was generated.

> **Definition 7: Canonical Act:** *An act in which the data is referenced using keys that are ids corresponding to fields on the parent persona, and which also contains a "UUID" record among its meta records that points to the DNAP persona UUID that can uniquely identify this parent persona.*

> **Definition 8: Annotated Act:** *An act in which the data is referenced using keys that are human-readable labels corresponding to fields on the parent persona, and which might or might not contain any "meta records".*

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Note that in canonical form, an act does not make it immediately obvious what the data means. Thus, to interpret the data held in such an act, it would be necessary to consult the parent persona specification – which one can obtain by fetching the persona referenced in the contained persona UUID of a valid act.

Given such a persona UUID, one can retrieve the necessary specification via a standard API call in which the UUID is used to tell the theatre to fetch the persona with the specified UUID value. An example of this is depicted in **Appendix A** for the UUID c6e47f79-afd9-4ebc-8753-3da5aa5f4cd7.

In some cases though, an act can be depicted in an annotated form (as opposed to the above "canonical form"), in which case, the Field IDs are replaced by the actual labels or names of the fields associated with each value in the original persona. This annotated form is for example what one sees when they access the act cache via the web (see **Appendix D**) or mobile (see **Appendix E**). Below is an example of the above canonical act as an annotated act:

Listing 4: Example Annotated Act

```
{
    "LOCATION":"1 Nuchwezi Road.",
    "WIND":"NORTH",
    "CLOUDS":"8",
    "RAINING":"YES",
    "GPS":"0.0903518,32.5429456",
    "OBSERVER":"P.",
    "OBSERVER_ID":"0700100000",
    "CACHE_TIMESTAMP":"2019-10-10T00:25Z"
}
```

3.5 The Theatre

This is the backbone or rather backend engine of DNAP. More formally;

> **Definition 9: Theatre:** *Any web server that satisfies the following requirements* ***at minimum****:*
>
> 1. *It can accept and store personas published via a studio, using a REST API*
> 2. *It can allow such published personas to be accessed or referenced from a compatible histrion using a REST API*
> 3. *It can accept and store canonical acts associated with these published personas via a REST API*
> 4. *It can accept and serve requests for canonical or annotated acts for known published personas when given the associated persona UUID, via a REST API.*

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

As you can tell from the above definition, the theatre acts as the "glue" that ties together the other components of the Dynamic Nuchwezi Architecture Platform. It is possible to use the studio and histrion without a theatre – for example, one could design a persona, and instead of publishing it, merely download and manually load it into a compatible histrion. But without a theatre, management and security of the tools and data thus generated becomes a nightmare! The key role of the theatre in DNAP is to make management of tools, data and the users that operate on them much more easy and straight-forward, and all this, done securely and via standard protocols.

As you shall see in the typical DNAP workflow (**Figure 5**), once a persona has been designed via the studio, one has the option to first publish it onto a theatre, before they use it. This is the recommended approach and highlights the first role played by the theatre on the DNAP.

Once a persona has been published — to a theatre that is, it is associated with a globally unique identifier, the "Persona UUID", which the theatre will know about, and using which, one can then fetch the persona specification by URL, so as to load it into a compatible histrion as a persona applet (see **Figure 4**). Such access is possible via a standard API of the form:

URL-1: https://<THEATRE_FQDN>/api/persona/<PERSONA_UUID>/

The above URL, which is accessible using HTTP GET, returns a JSON document that is the specification of the persona referenced by the specified PERSONA_UUID. An example of such a specification is in **Appendix A**.

The third core role of a theatre is that it allows data captured via these personas, on compatible histrions, to be collected and stored in one place for later use or reference. This is the reason there is a "theatre_address" parameter on each persona specification; at design time, the author of a persona needs to specify how data acquired via the persona shall be transported (the reason there is a "transport_mode" parameter) and where that data shall be submitted to – the *theatre_address*, which for the typical case (in which acts are submitted using HTTP POST), is the URL of a REST endpoint on a compatible theatre. For example, in the reference implementation of the DNAP, the default Theatre Address for personas built via the reference studio implementation is of the form:

URL-2: https://<THEATRE_FQDN>/api/act/create/

Where <THEATRE_FQDN> is 'chwezi.tech' by default. But it is important to note that not all personas need specify a URL for a theatre address; where the transport_mode parameter is specified as "EMAIL" or "SMS" for example, this parameter expects an email address or phone number respectively. In this case, we can expect these to be specified as valid URIs using the "mailto:" or "tel:" schemes respectively – though, in the default implementation of the histrion, the scheme part can be eliminated, so that this parameter merely takes in a URN [SK17].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

Finally, the theatre should be able to accept HTTP GET requests for, and serve any data submitted to it for the specified persona, given the persona UUID. In general, such a request involves making a call to a REST endpoint on the theatre, using a URL of the form:

URL-3: https://<THEATRE_FQDN>/api/persona/<PERSONA_UUID>/acts/

Which returns all currently stored data for the persona specified using PERSONA_UUID as annotated acts. Alternatively, there is

URL-4: https://<THEATRE_FQDN>/api/persona/<PERSONA_UUID>/racts/

which returns the same, but as canonical acts. Note that in either case, the result of performing such a request on the theatre returns a JSON array, whose only contents are the acts or which is empty – in the case where no data exists yet for the given persona or where access to the act stream is restricted without proper authentication.

This mechanism allows data acquired via DNAP to be immediately consumable from almost any standard data processing interface – web-browsers; text-editors; command line tools such as wget, curl or jq; spread-sheet programs such as Microsoft's Excel or Google's Sheets, or even professional data analysis and business intelligence platforms such as PowerBI, Tableau and others.

By requiring that the theatre conform to the above specification, it then makes building of data acquisition and data sharing tools via the DNAP much more straight forward, and standardizes such a process, so that once a tool is published, it is immediately and readily obvious how data clients shall reference and or access data associated with the tool, and that this data-access interface can be relied upon without worrying about how or where the actual data storage happens.

The theatre REST API abstracts the underlying web and database technologies required to actually implement a robust data storage and retrieval platform, and given that these features come straight out of the box for those using a standard DNAP instance, developing and utilizing data apps becomes so simple, even for non-technical users or those with no experience building client-server, web-based or database-powered software. This later point is one of the original key motivations for the Dynamic Nuchwezi Architecture Platform.

In relation to the above, it is important to note that for purposes of keeping things simple, DNAP further standardizes how access to the data acquisition and data analytics interfaces of a given persona are to be done. This especially applies to web-browser-oriented use-cases. Thus, once published to a theatre, and once the persona UUID is known, then access to the web histrion and diviner of the associated persona is possible via the standard URI forms:

URL-5: https://<THEATRE_FQDN>/persona/<PERSONA_UUID>/histrion/

URL-6: https://<THEATRE_FQDN>/persona/<PERSONA_UUID>/diviner/

For the histrion and diviner respectively. The histrion has already been discussed in the previous sections, next we shall consider the diviner.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

3.6 The Diviner

> **Definition 10: Diviner:** *A general purpose app that can parse and interpret persona source code from which it then renders act stream analytics applets.*

With reference to this definition, we note that like the histrion, the diviner is a general-purpose app, except that instead of rendering data acquisition interfaces, it renders data analysis interfaces for the associated persona(s).

The motivation for the diviner is simple; for the typical use case, a user wants to collect data to later analyze it. Thus, instead of merely offering the user a means to collect and share data - which the histrion and theatre does, the diviner further offers them a means to immediately be able to analyze data collected via those mechanisms, in the simplest way possible.

As of this writing, the only diviner implementation available runs on the web, and as such, is accessible via a web browser (like the studio, theatre and web histrion components of DNAP). In the previous section, we have already seen how when given a published persona's UUID, and the theatre via which it is published, it becomes possible to access its diviner (refer to **URL-6**). However, like the web histrion, there exists a standalone diviner implementation [Lut20a] that one can access directly, and then, equipped with a valid persona URI, one can load and analyze incoming or existing data associated with the persona.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

4 How DNAP Works

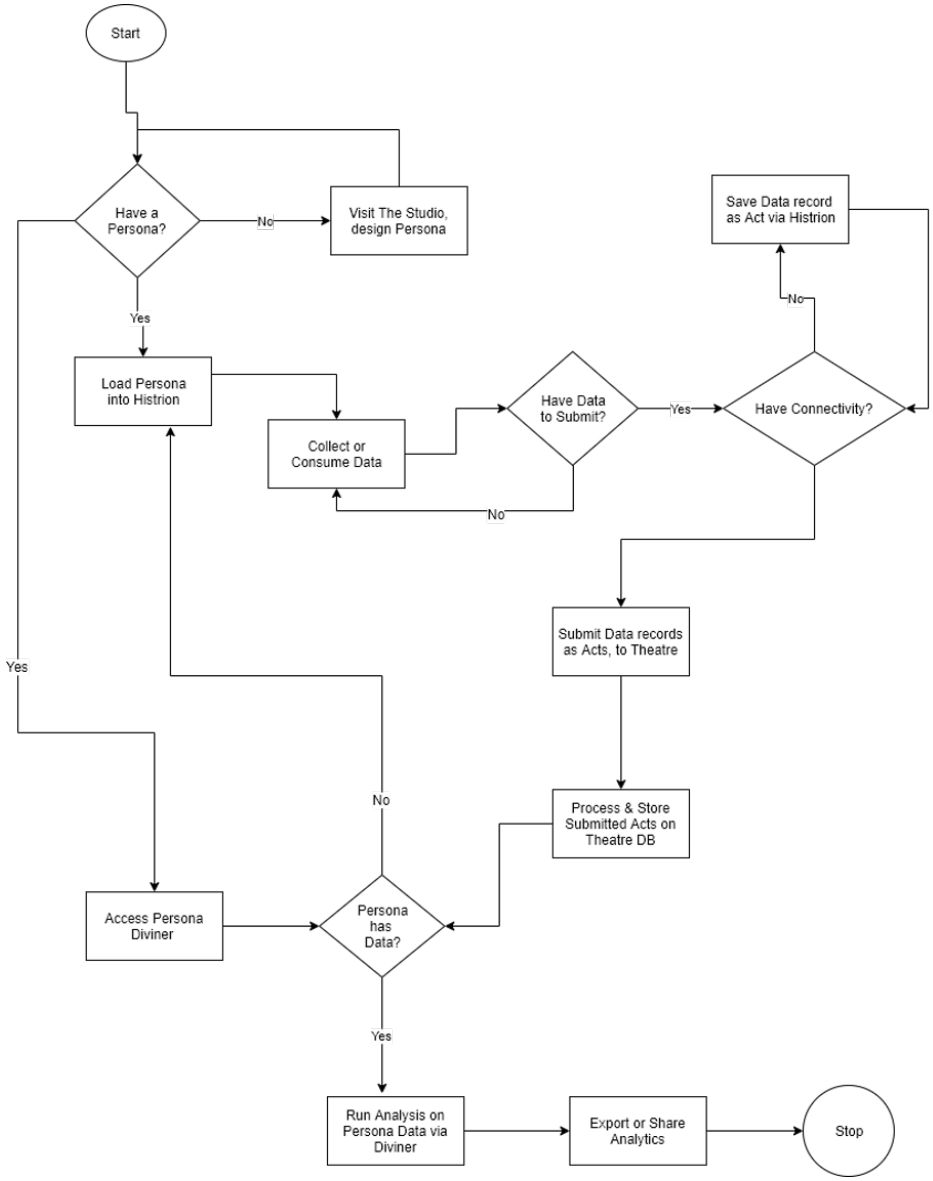

Figure 5: DNAP Work Flow

As indicated in the introduction, we shall limit our evaluation of DNAP to the data engineering case for now. Typically, an end-to-end data engineering task involves designing a data collection tool such as are traditionally called “forms”,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

publishing it somewhere, capturing data with it, storing the data, then later analyzing or sharing the collected data.

With DNAP, this basic, simple storyline does not change, but the details — the "subplots", are merely further simplified and standardized. In summary, here is how the end-to-end data engineering task proceeds via DNAP:

1. Design and compose your persona via a studio.
2. Publish the persona to a theatre.
3. Load the persona into a histrion (could fetch the persona from the theatre or use a downloaded copy from the studio).
4. Activate the persona on the histrion.
5. Interact with the persona – e.g. capture data using the persona.
6. Verify the data (automatic step upon submission or saving).
7. Save the data (useful when working offline or in areas of poor connectivity)
8. [optionally] Edit the saved data (currently only for the mobile histrion via the "clone" feature [Lut19])
9. Submit the data to the theatre (requires an active data connection).
10. Visit the diviner instance of your persona (link shared via studio at persona publication time or via the theatre after publication — refer to **URL-6**).
11. Fetch, browse and analyze any data found on the persona's act stream endpoint (see **URL-3** and **URL-4**)
12. Export, download or share visuals and any other information generated from this analysis.

Note that in practice, it is possible to kick-start this process by starting from an already existing persona instead of creating one from scratch. This facility is in-built on the reference implementation of the studio, and it allows one to kick-start their project in one of several ways:

1. You can clone existing personas via their resource URLs. Request URL follows the pattern:

   URL-7: https://<STUDIO_FQDN>/*?purl=<PERSONA_URI>*

2. You can clone an existing persona by finding and loading its *.persona file (e.g. one previously downloaded via a studio or a mobile histrion)
3. You can clone a persona whose specification you have (e.g. copying and pasting the code in **Appendix A** into a studio)
4. You can clone a persona by selecting it from the list of template personas shown under the "Templates" section on a studio.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

5 DNAP's Novelty

DNAP differs from some of its predecessors and contemporaries in the following ways especially;

1. Where the central logical structure of other technologies and platforms is the "form", for DNAP, it is something like a form, but with more capabilities than a typical data acquisition tool; we refer to this new structure as the "persona", and we have formally defined what it is in **Definition 1**.

2. If we contrast DNAP's persona with "apps" or "tools" in other platforms or systems of a similar kind, then we see that DNAP simplifies development — in this case persona development, by making it much easier to start a new project from existing ones thus:

    - One-click creation of a new persona based on existing persona templates
    - Creation of a new persona by cloning an existing one accessible by its repository URL
    - Creation of a new persona by importing the code of a previously downloaded one.

3. Ability to craft a persona directly in code from any editor: whereas other platforms supporting a visual programming paradigm allow simple authoring of tools in a manner akin to DNAP studio's drag-and-drop, the option to go manual and code a tool from scratch by hand is often lacking — for example, all the platforms reviewed in **Table 1** except DNAP don't offer this option. This feature is aimed at experienced users or engineers that might need to author and use tools while on the move with only access to a histrion for example. DNAP makes it possible to use both visual or textual programming for persona development.

4. The speed and ease with which deployment of a persona can be done across many devices:

    - Deployment can be done by downloading the designed persona (encoded as a DNAP persona file - i.e *.persona), which can then be shared via mail, social media, or any file-transfer method, so as to later be run on mobile or web via a histrion. Meaning, personas are exportable and thus shareable outside the DNAP ecosystem.
    - Deployment can be done using "channels", whereby the persona is published under a specific keyword, that is then used by subscribers to automatically discover and fetch the persona (and any other related ones) from an online repository such as a theatre. This feature brings to DNAP the concept of automated data service discovery as applied in Service Oriented Architecture (SOA).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

- Deployment can be done using QR Codes; in this case, a QR Code is generated every time a persona project is published via the studio or when it is accessed via the persona manager on a histrion or theatre, and this QR Code image (typically a PNG) is all that needs to be shared for the persona to be distributed to others. The persona QR Code can be scanned using a histrion, from anywhere, which then fetches, installs and runs the target persona as specified by the persona QR Code. This allows for quick and effortless sharing of persona-software across devices without actual transfer of the persona files or configuration of anything on the side of the users.

5. Personas run on mobile as native mobile apps; the histrion turns any persona into a native mobile app, complete with custom themes and the ability to create custom launchers that make using personas feel like using ordinary mobile apps than traditional form widgets that other platforms such as Open Data Kit (ODK) [Har+10] offer.

6. Additionally, these personas can function offline; for data acquisition, storage and dissemination especially (recall that DNAP offers data transport via SMS), though data analysis and some types of data dissemination (HTTP POST/GET and EMAIL) require connectivity to function.

7. Data Resilience —- apart from supporting offline rendering of personas on mobile and web, the mobile histrion supports a feature called "Sticky-Acts", which can be manually or automatically turned-on or off for each cached persona, and which feature, when enabled, lets the app automatically save a persistent, local (on-device) copy of any data created by the user via that persona, regardless of whether the data was posted to the online repository yet or not. In the reference implementation, the histrion never deletes sticky-acts unless the user manually triggers this. This feature offers a simple, client-side data-recovery mechanism that can be very useful in the absence of a reliable data connection between histrions and their configured theatre or if the theatre has some data deleted or lost, and the act stream needs to be created or updated anew.

8. Personas can explicitly be linked to each other; for example, it is possible to design a data acquisition tool that includes a mechanism to hot-load another related or dependent tool on-demand from the histrion. That way, it becomes possible to build sophisticated, multi-step forms or wizards, or even implement hierarchical documents akin to websites directly on DNAP.

9. Personas support simple relational data models; for example, it is possible to design a persona applet that includes a menu field whose items are auto-populated based on the posted data (the "act stream") of yet another another persona. This ability to reference fields from other personas is based on the data APIs automatically generated by the theatre for every published persona, and which APIs offer a means for clients to query,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

filter, and apply basic transformations to the act stream via a simple and secure RESTful mechanism.

10. DNAP tools are not limited to data acquisition form-type apps only, no. It is possible to design a persona whose purpose it is to only show, but not collect data. This can include showing text, images, playing back audio or video, displaying the contents of a website or even displaying the data from the same or another persona. With this kind of capability, DNAP further sets itself apart from other similar technologies in that it can be used to build a wider category of apps outside of traditional data forms — say, e-magazines, periodicals, marketing apps, e-learning apps, music players, and more. And further, if well designed, the content of these apps can be cacheable so that users continue to view the integrated media even while offline! On the other hand, the content can be designed to be dynamic such that content changes between persona reloads, based on client-side interactions or other conditions such as time, thus making DNAP a good candidate for building multi-media, interactive solutions too.

11. Feature Completion as a PAAS: without relying on any external or 3rd-party tools, the DNAP suite includes all the elements required to perform typical data engineering tasks end-to-end; design, host, distribute and run data collection and analysis tools from one place, and all this, without requiring the user to configure any servers — essentially, DNAP makes possible the implementation of a Platform-As-A-Service (PAAS), so users can build and use apps on the platform without getting into any of the technicalities.

12. Further, DNAP's data analysis tools allow for the design and use of persistent analytics dashboards useful in long-term data monitoring, management and reporting scenarios.

6 DNAP applied to Data Analysis

When it comes to data engineering, the main tasks involve being able to build mechanisms for performing the data acquisition, data aggregation and finally data analysis. This trio we have already explored in the previous sections when looking at the various components of the DNAP and how it works, however, let's delve deeper into the last item, which is arguably the most important.

As mentioned earlier, DNAP deals with acts (see **Definition 6**), which are a form of structured data. There are many ways structured data can be analyzed, but generally, the choice of analysis method depends on the structure of the data, the domain or purpose of the data and the preferences or skills of the analyst. However, for DNAP's current diviner implementation [Lut20a], analysis is of an exploratory kind by default – what the formal literature refers to as Exploratory Data Analysis (EDA).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

That DNAP gives focus and priority to EDA is justified by the importance of this form of analysis in most, if not all early phases of scientific research and analysis. In their seminal work on the subject, Yu [Yu77] stresses that EDA is a very important step which takes place after feature engineering and acquiring data, and that it should be done before any modeling. This, they indicate, is because it is very important for a data scientist to be able to understand the nature of the data without making assumptions. Further, the purpose of EDA is to use summary statistics and visualizations to better understand data, and thus find clues about the tendencies of the data, its quality and to formulate assumptions and the hypothesis of the subsequent analyses.

In this regard, it comes as no surprise that DNAP's diviner gives priority to EDA by making available the following analytical capabilities on any data collected using the platform:

1. Generate a basic tabular projection of the data
2. Generate an advanced tabular projection of the data – includes pagination, sorting, search and filtering
3. Generate a line chart from the data – requires than one of two chosen fields include a numerical type (or where both are categorical, a numerical transform such as COUNT be applied)
4. Generate a time series chart of the data - requires than one of two chosen fields include a timestamp type (note that by default, all acts generated via the reference histrions include a timestamp in the CACHE_TIMESTAMP field for this purpose)
5. Generate a scatter plot from the data
6. Generate a pie chart from the data
7. Generate a bar chart from the data
8. Generate a JSON projection of the data – useful where the analyst wishes to apply some basic transforms before exporting the data out for further analysis by an algorithm.

Any of the above are basically achieved by taking two or more fields specified via the "Domain" and "Range" selectors on the diviner interface, and then generating the target visualization or data projection when the "RENDER" button is clicked (see **Appendix F**).

Apart from allowing one to perform basic EDA tasks, the diviner also makes it easy to export data out of DNAP for further analysis in other, more expert-oriented tools. In this regard, the following extra capabilities are supported:

1. Ability to export the data as comma separated values (CSV)
2. Ability to export the data as a PDF table

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

3. Ability to export the currently active visual as a PNG image.

But, these features are all good, except for one short-fall; any analysis done during a particular EDA session (such as that in **Appendix F**), will be lost, and will need to be re-run if the diviner session ends — say if the browser in which the analysis is being done gets closed or reloaded. Thus, to mitigate this, and in order to allow the analyst to be able to run and view multiple analyses on the same dataset at once, a dashboard mechanism was also built into the DNAP diviner.

In summary, this dashboard mechanism allows the analyst to:

1. Add one or more analysis panels to a dashboard.
2. Remove an analysis panel from an existing dashboard
3. Name and save the dashboard for later preview — allows same EDA session to be auto-run even across browser reloads or restarts.
4. Load a previously saved, named dashboard on-demand — the dashboard needs to be loaded against the same persona act stream as was originally used when designing it.

For an example of what a dashboard looks like on the DNAP diviner, check **Appendix G**.

7 Potential Users of DNAP

In brief, DNAP makes it easy to design, deploy and manage mobile and web tools for structured data acquisition and analysis, as well as tools for information dissemination or combinations of both. But more specifically, the platform targets the following categories of users and use-cases:

- Monitoring and Evaluation teams, Data Surveyors and Data Analysts: these can be business owners, managers or senior staff in need of simple, continuous, and reliable monitoring and evaluation of their business processes via mobile or web, in and out of office.
- Data Analysts that already have data pathways from legacy systems, but who need to extend their data footprint by adding new data acquisition channels in a simple, but efficient manner — in particular, without having to manually code new apps and without the need to hire a programmer to build the extensions for them.
- NGOs and other large organizations whose optimal performance relies on having quick and reliable access to data and insights about internal or external metrics, and which data they need to collect themselves or via third-parties on-demand, with low ICT overheads, and especially while keeping such operations low on budget.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

- Researchers at all levels — whether it is students wishing to document school lab experiments, professors collecting a little extra data for their esoteric projects or industry experts in need of a quick and effective way to source for actionable data about difficult processes; whenever one has a need to put together a form for collecting data, and if they need to use something better than paper or a spreadsheet for this, then DNAP offers a simple, cost-effective and technically reliable option.

- Data Hobbyists — it could be friends of democracy, that just wish to crowd-source for opinion using easily distributable polls; it could be someone in need of documenting and analyzing their own eating habits for example ("self-quantification") using an app they designed themselves, etc.

- Data Disseminators or Publishers: Using the information sharing capabilities of personas – showing images, video, audio, text, web pages, etc. It is possible to design and distribute information products in the form of e-magazines, periodicals, e-books, podcasts and more, over DNAP, in a way that combines the ability to share information and obtaining feedback about it, and all this greatly simplified relative to the other methods available today.

8 Known Limitations of DNAP

As with any software or technology platform, there are both structural and technological limitations to be found in DNAP. Among these, the most notable ones include:

- DNAP's diviner, which offers data analysis functionality, can currently only operate on data in the form of acts.

- The web histrion [Lut20b] can't reliably support all the field types that the mobile histrion currently supports. For example, using GPS, Camera and Barcode field types meets with challenges across various browser types and operating systems, which is not the case with the native mobile histrion [Lut19].

- If a malicious entity discovered the UUID of a persona not meant for public, anonymous access, they still can go ahead to craft a persona QR Code with which access to the persona and its dataset can be obtained without the owner of the tool discovering this. There is currently a limited mechanism to prevent access to the dataset of personas requiring an access token for their data APIs, but this feature is only configurable by those users with access to a persona's administration pages on its associated theatre - which in the reference implementation requires an invite or administrator account [Lut20a].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

- DNAP's studio and histrions can't yet handle such field types as input via OCR, and though support for a sensor-input such as GPS is currently supported, other obvious types such as temperature, humidity, luminosity, and the like aren't yet supported.
- Though ability to populate data-driven field types such as the Menu field-type can leverage remote, online data sources at runtime, this capability is currently restricted to data specifically formatted as a flattened JSON array of strings or numbers, and this capability only implemented in the mobile histrion.
- Data submitted to the online data repository, the theatre, can only be editable by users with administrator access to the associated persona's administration pages, and this access is not readily obtained via the persona authoring tool - the studio. However, those using the mobile histrion can still edit data they locally hold on their device as "Saved Acts".
- Currently, there isn't much analytics capabilities offered on DNAP's diviner beyond EDA. There is a simple regression analysis capability that was introduced for pairs of numeric fields, but it isn't well evolved and can't handle categorical or temporal data types yet.

9 Conclusion

All in all, the Dynamic Nuchwezi Architecture Platform offers a new way for both professionals and amateurs to design, publish, share and exploit web and mobile tools for data acquisition and dissemination in a straight-forward, much simplified way. By championing a simple, browser-based visual coding paradigm as the default way to design and deploy these tools, DNAP readily realizes the modern idea of "Citizen Programming" that is of growing interest in the software engineering industry [Mas16], and thus offers a much needed solution in a world that increasingly needs to put the power to perform automation in the hands of as many people as possible.

It is desired that DNAP becomes a global technology standard, and it definitely can be, even as it is currently specified and implemented, however there is still room for further improvement so as to make it ideal; the design and implementation could be further enhanced to support more use-cases in the data acquisition, data dissemination and data analysis contexts. The objective being to make DNAP a fully self-contained solution for those performing real-world data-driven work hinged on collecting or consuming data and producing analyses or other apps as result — this could be in domains such as data science, business intelligence or basic data engineering in the context of business automation. Finally, it is desired that DNAP continue pushing the edge in the creation of data tools capable of serving a useful purpose in totally offline scenarios and especially via mobile.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

References

[Yu77] Chong Ho. Yu. "Exploratory data analysis". In: *Methods* 2 (1977), pp. 131–160.

[Har+10] Carl Hartung et al. "Open data kit: tools to build information services for developing regions". In: *Proceedings of the 4th ACM/IEEE international conference on information and communication technologies and development*. 2010, pp. 1–12.

[MB13] Marla Mallette and Diane Barone. "On using Google forms". In: *The Reading Teacher* 66.8 (2013), pp. 625–630.

[DIP15] Mireille Djenno, Glenda M Insua, and Annie Pho. "From paper to pixels: using Google Forms for collaboration and assessment". In: *Library Hi Tech News* 32.4 (2015), pp. 9–13.

[Mas16] Dave Mason. "Ubiquitous Citizen Programming". In: *Procedia Computer Science* 98 (2016), pp. 169–173.

[Bra17] Tim Bray. The JavaScript Object Notation (JSON) Data Interchange Format, 2017. DOI: 10.17487/RFC8259.

[SK17] P Saint-Andre and J Klensin. "Uniform Resource Names (URNs)". In: *Internet Engineering Task Force (IETF), RFC* 8141 (2017). DOI: 10.17487/RFC8141.

[AW19] H Andrews and A Wright. "JSON schema: A media type for describing JSON documents". In: *draft-handrews-json-schema-02 ed.* 2019. URL: https://json-schema.org/latest/json-schema-core.html.

[Dat19] Datascope. *Datascope Features*. 2019. URL: https://www.mydatascope.com/features.

[Goo19] Google. *Google Forms*. 2019. URL: https://www.google.com/intl/en/forms/about/.

[Kin19] Kinto. *Kinto Formbuilder*. 2019. URL: https://github.com/kinto/formbuilder.

[Lut19] Joseph Willrich Lutalo. *DNAP: Data via Nuchwezi Architecture*. Google Play Store, 2019. URL: https://play.google.com/store/apps/details?id=com.nuchwezi.historion.

[Zoh19] Zoho. *Zoho Survey. Zoho Corporation.* 2019. URL: https://www.zoho.com/survey/.

[Lut20a] Joseph Willrich Lutalo. "The DNAP Data Diviner". In: (2020). Last accessed on 2020-05-01. URL: https://diviner.nuchwezi.com/.

[Lut20b] Joseph Willrich Lutalo. "The Web Histrion". In: (2020). Last accessed on 2020-05-01. URL: https://histrion.nuchwezi.com/.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

A Example Persona Specification: NJURA

Originally retrieved via the online reference implementation of a DNAP-compatible theatre[1] , this persona describes a tool used in crowd-sourcing location-tagged updates about weather, and especially occurrences of rain. The complete persona specification is here reproduced for your reference:

Listing 5: Example Persona Specification: NJURA

```
{
        "fields": [{
                        "field_type": "text",
                        "cid": "c2",
                        "pattern": "",
                        "required": true,
                        "label": "LOCATION",
                        "field_options": {
                                "size": "small"
                        }
                }, {
                        "field_type": "dropdown",
                        "cid": "c10",
                        "pattern": "",
                        "required": true,
                        "label": "WIND",
                        "field_options": {
                                "options": [{
                                                "checked": false,
                                                "label": "NORTH"
                                        }, {
                                                "checked": false,
                                                "label": "NORTHEAST"
                                        }, {
                                                "checked": false,
                                                "label": "EAST"
                                        }, {
                                                "checked": false,
                                                "label": "SOUTHEAST"
                                        }, {
                                                "checked": false,
                                                "label": "SOUTH"
                                        }, {
                                                "checked": false,
                                                "label": "SOUTHWEST"
                                        }, {
                                                "checked": false,
                                                "label": "WEST"
                                        }, {
                                                "checked": false,
                                                "label": "NORTHWEST"
                                        }
                                ],
                                "description": "Direction from which the wind is blowing the most.",
                                "include_blank_option": false
                        }
                }, {
                        "field_type": "dropdown",
                        "cid": "c22",
                        "pattern": "",
                        "required": true,
                        "label": "CLOUDS",
                        "field_options": {
                                "options": [{
                                                "checked": false,
                                                "label": "0"
                                        }, {
                                                "checked": false,
                                                "label": "4"
                                        }, {
                                                "checked": false,
                                                "label": "8"
                                        }
                                ],
                                "description": "0 = completely cloudless sky\n8 oktas = sky is fully clouded",
                                "include_blank_option": false
                        }
                }, {
                        "field_type": "radio",
                        "cid": "c18",
                        "pattern": "",
                        "required": true,
```

[1] https://chwezi.tech/api/persona/c6e47f79-afd9-4ebc-8753-3da5aa5f4cd7/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2


```
                        "label": "RAINING",
                        "field_options": {
                                "options": [{
                                                "checked": false,
                                                "label": "YES"
                                        }, {
                                                "checked": false,
                                                "label": "NO"
                                        }
                                ],
                                "description": "seen any rainfall from where you are right now?"
                        }
                }, {
                        "field_type": "devicegps",
                        "cid": "c6",
                        "pattern": "",
                        "required": true,
                        "label": "GPS",
                        "field_options": {}
                }
        ],
        "app": {
                "theatre_address": "https://chwezi.tech/api/act/create/",
                "name": "NJURA",
                "color": "#545465",
                "transport_mode": "POST",
                "uuid": "c6e47f79-afd9-4ebc-8753-3da5aa5f4cd7",
                "brand_image": "",
                "channel": "LABS",
                "description": "Track weather changes for later analysis"
        }
}
```


B EXAMPLE OF A PERSONA RUNNING ON THE DNAP WEB HISTRION: NJURA

Figure 6: Web Persona Applet: Njura

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

The above screenshot is of the same persona defined in **Appendix A**, as rendered on a web based histrion hosted on the reference DNAP theatre[2].

[2]https://chwezi.tech/persona/c6e47f79-afd9-4ebc-8753-3da5aa5f4cd7/histrion/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

C EXAMPLE OF A PERSONA RUNNING ON THE DNAP MOBILE HISTRION: NJURA

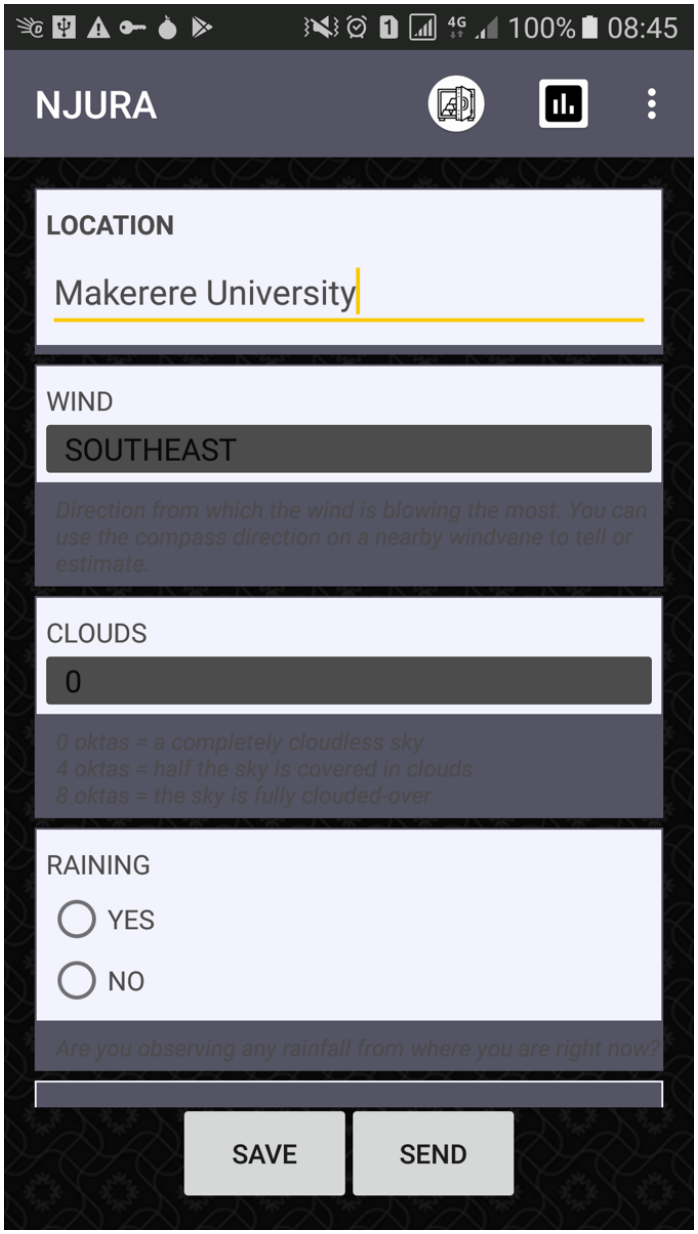

Figure 7: Mobile Persona Applet: Njura

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

D EXAMPLE OF AN ANNOTATED ACT ON THE DNAP WEB HISTRION'S ACT CACHE

Figure 8: Web Persona Acts: Njura

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

E EXAMPLE OF AN ANNOTATED ACT ON THE DNAP MOBILE HISTRION'S ACT CACHE

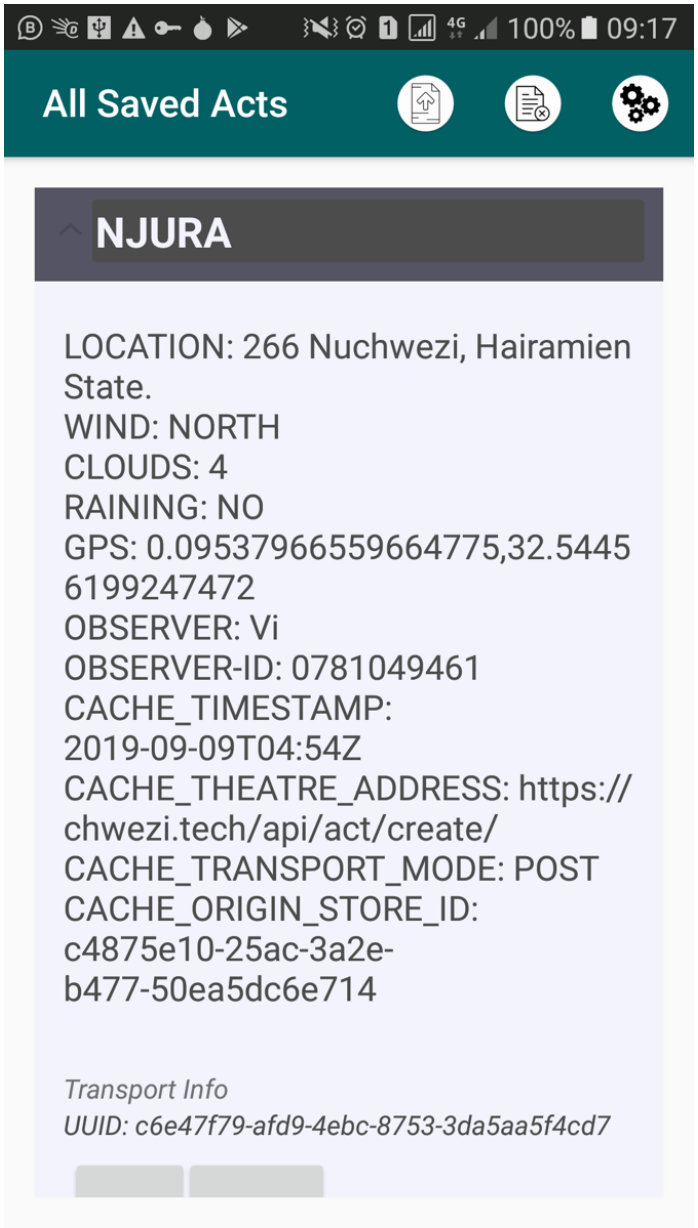

Figure 9: Mobile Persona Acts: Njura

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

F EXAMPLES OF AN EDA SESSION ON DNAP'S DATA DIVINER

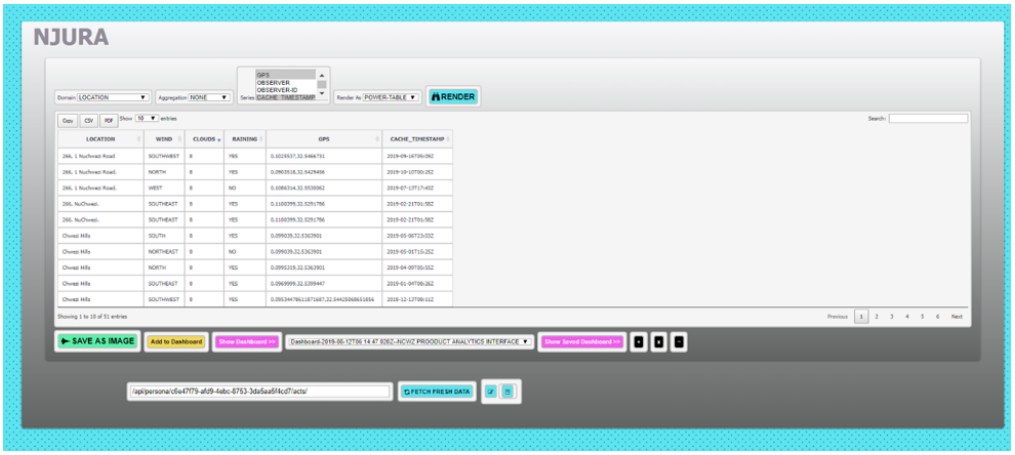

Figure 10: Diviner Applet: Njura data table

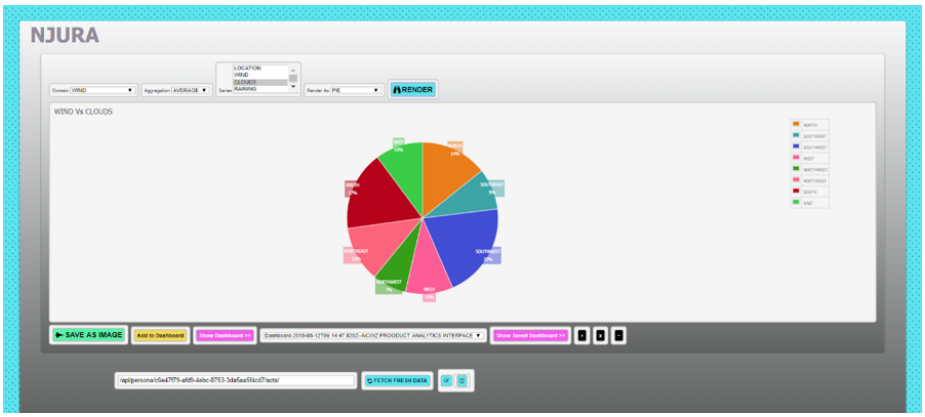

Figure 11: Diviner Applet: Njura basic categorical data visualization

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

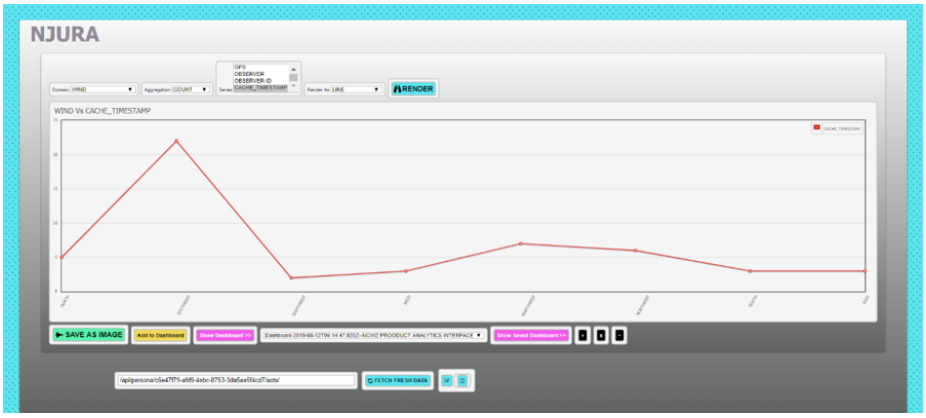

Figure 12: Diviner Applet: Njura basic time-series data visualization

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 May 2020 doi:10.20944/preprints202005.0207.v2

G EXAMPLES OF AN EDA DASHBOARD SESSION ON DNAP'S DATA DIVINER

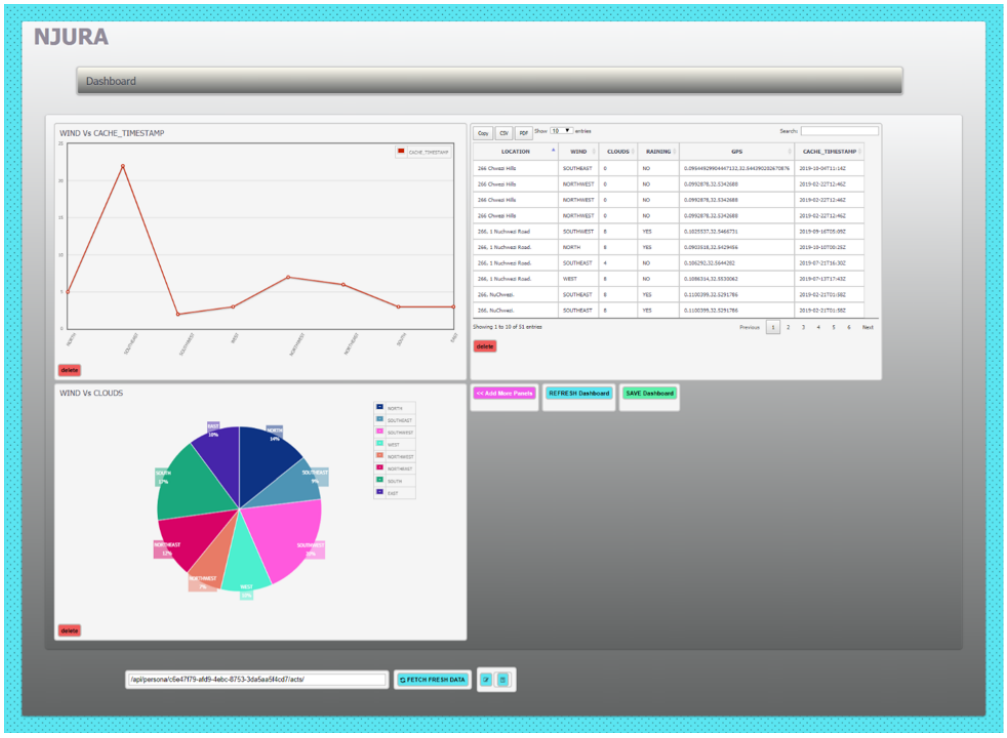

Figure 13: Diviner Applet: Njura data analysis dashboard