
CoronaHackathon

wir

Mar 23, 2020

CONTENTS

1	Introduction	3
2	Application	5
2.1	Questionnaire Generator	5
2.2	Border Crosser Webapp	8
2.3	Border Police Website	9
2.4	Backend: Connecting the Pieces	10
	Python Module Index	13
	Index	15

As part of the [WirvsVirus](#) hackathon we join the fight against COVID-19.

The last couple of weeks were defined by the spread of the virus around the world. Europe, in particular, rapidly became the epicenter of the current epidemic. As a consequence, borders throughout Europe were closed in a rush leading to heavy traffic jams. Europe's free and borderless market, risks being halted due to merchants, commuters, and truckers being stuck attempting to cross a border.

For this weekend we chose to **hack** together a prototype app that allows officials to provide questionnaires digitally, enabling border crossers to answer questions in advance and reduce the work of border policemen. Checkout our video on [YouTube](#) to see what we have achieved!

INTRODUCTION

Consider the following user story:

A truck driver delivering goods throughout Europe

On his journey, the truck driver travels from country A to B but does not speak the language spoken in country B fluently.

How can the time of the border crossing be minimized?

A policeman of country B at the border will want to know the truck driver's identity, his destination, and reasons for travel. Most of the time will be lost communicating and answering questions while overcoming the language barrier.

Note: What if the truck driver could answer the questions in advance, **in his own language**?

The time it would take for the policeman to verify the answers would be only a fraction of the time it currently takes and seriously reduces traffic jams all over Europe's closed borders.

Fig. 1: Mock Up of the questions a person crossing the border from Germany to Switzerland might answer using our application.

APPLICATION

We took this scenario as a guideline for our application and decided to write the following tools:

- A questionnaire generator allowing officials to upload and distribute questionnaires to people who want to cross their border.
- A client webapp that allows anyone to answer the questions in their own language and generate a QR-Code containing the answers as a URL.
- A website containing the answers in the policeman's language.

Therefore allowing policemen to get answers to all questions they are supposed to ask just by scanning a QR-Code!

In the following the details of our implementation are documented.

2.1 Questionnaire Generator

The questionnaire generator is a simple qt based python program that allows officials to generate questionnaires in the shape of a JSON File. The JSON File contains questions and language information.

2.1.1 project.dict_generator module

Summary

Functions:

<i>add_lang_dialog</i>	Add language dialog to left window.
<i>cancel_inspect_mode</i>	
<i>change_lang_inspect_mode</i>	Go into inspection mode.
<i>change_option</i>	Change view.
<i>close_application</i>	
<i>create_new_dict</i>	Initialize a new dict.
<i>create_table</i>	Create a new table.
<i>fill_new_language</i>	Fill new language adding it to the dict_state.
<i>finish_new_language</i>	Forward button when adding new language.
<i>handle_table_click</i>	
<i>new_dict_begin_dialog</i>	Open dialog to add a new dictionary and start with entering a new language.
<i>new_dict_dialog</i>	Switch to new dict dialog window.
<i>next_new_language</i>	Forward button when adding new language.

Continued on next page

Table 1 – continued from previous page

<code>open_dialog</code>	Open a file dialog to read json dict from file.
<code>prev_new_language</code>	Back button when adding new language.
<code>save_dialog</code>	Open a file dialog to save json dict to file.
<code>set_left_window</code>	Go into inspection mode.
<code>update_lang_inspect_table</code>	Update language table.
<code>updated_table</code>	Updated table.
<code>write_to_dict</code>	Write

Reference

`project.dict_generator.save_dialog()`
Open a file dialog to save json dict to file.

`project.dict_generator.open_dialog()`
Open a file dialog to read json dict from file.

`project.dict_generator.add_lang_dialog()`
Add language dialog to left window.

`project.dict_generator.new_dict_dialog()`
Switch to new dict dialog window.

`project.dict_generator.new_dict_begin_dialog()`
Open dialog to add a new dictionary and start with entering a new language.

`project.dict_generator.create_table()`
Create a new table.

`project.dict_generator.updated_table()`
Updated table.

`project.dict_generator.fill_new_language()`
Fill new language adding it to the dict_state.

`project.dict_generator.prev_new_language()`
Back button when adding new language.

`project.dict_generator.next_new_language()`
Forward button when adding new language.

`project.dict_generator.finish_new_language()`
Forward button when adding new language.

`project.dict_generator.handle_table_click(row, column)`

`project.dict_generator.set_left_window(idx)`
Go into inspection mode.

`project.dict_generator.create_new_dict()`
Initialize a new dict.

`project.dict_generator.change_option()`
Change view.

`project.dict_generator.change_lang_inspect_mode(row, column)`
Go into inspection mode.

`project.dict_generator.update_lang_inspect_table()`
Update language table.

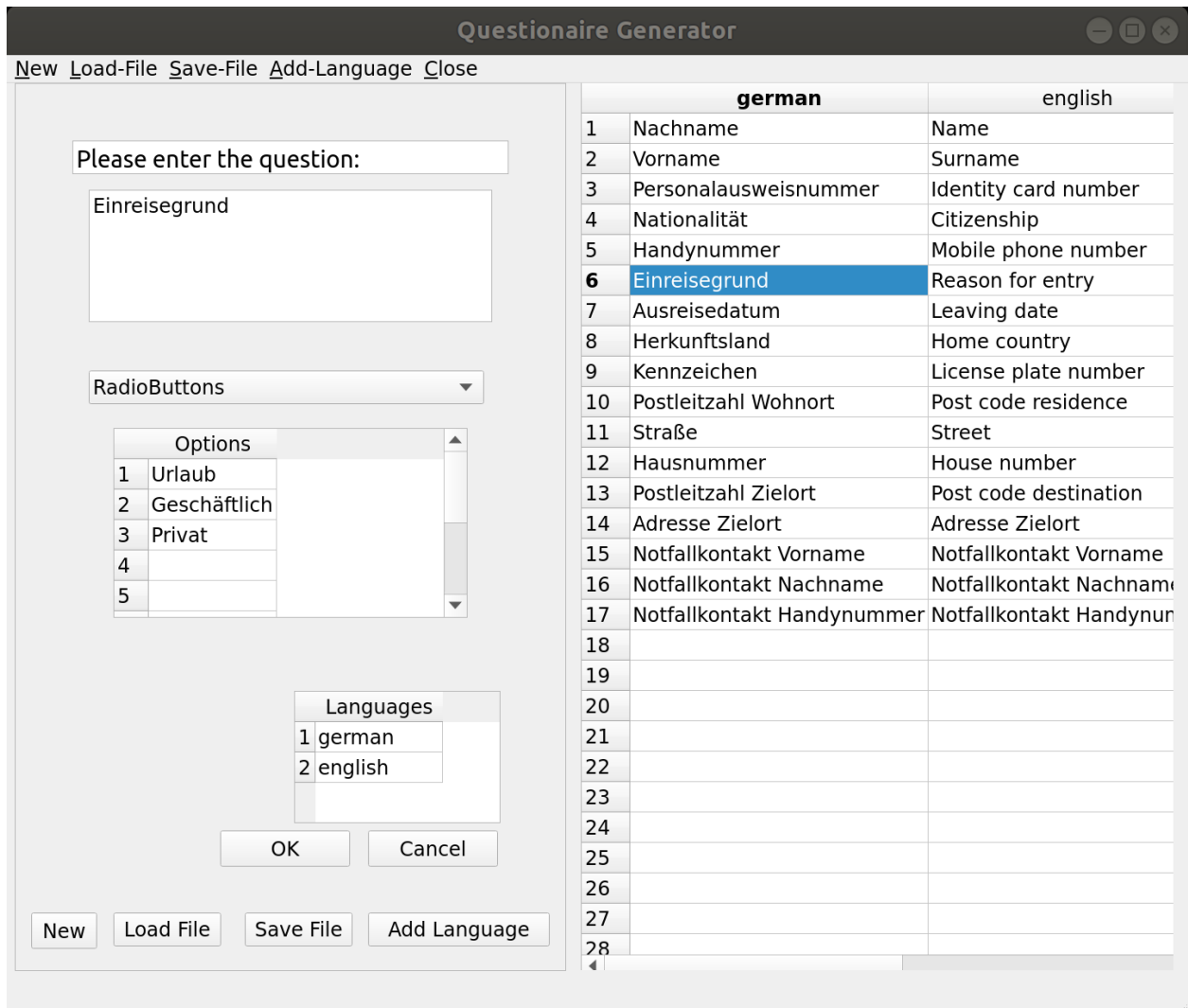


Fig. 1: Image of the Questionnaire Generator

```
project.dict_generator.cancel_inspect_mode()  
project.dict_generator.write_to_dict()  
    Write  
project.dict_generator.close_application()
```

2.2 Border Crosser Webapp

The border crosser's web app consists of only one website that, once opened, downloads the correct questionnaire and allows us to generate the resulting QR-Code without any further internet connection.

Our current state looks like this:

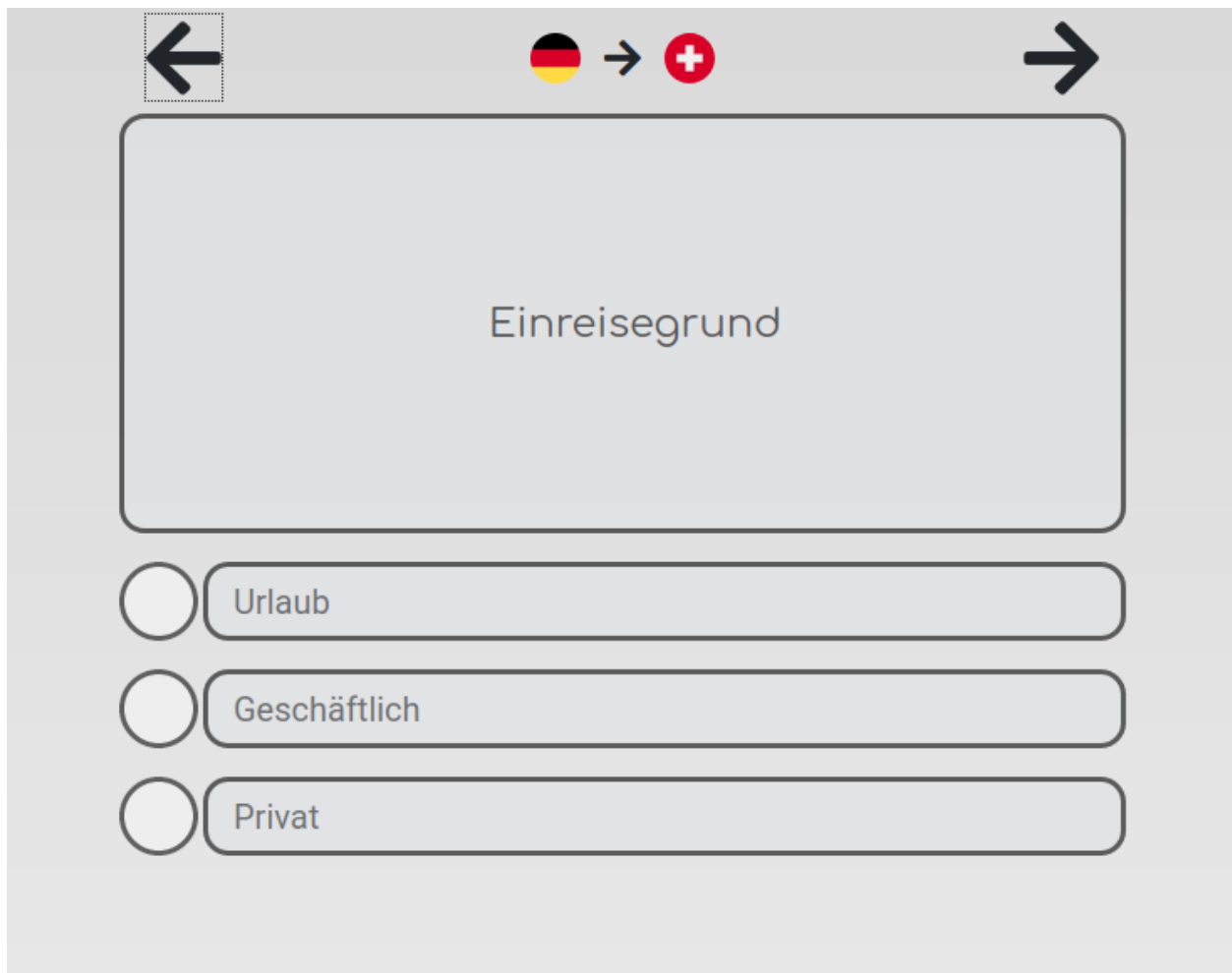


Fig. 2: Image of the Border Crosser App

This is how we imagine the app to be used on a mobile device:

2.3 Border Police Website

Encoded in the QR code is an URL containing all answers of the border crosser. The border police can then simply scan the QR code and open the website to see the answers on her/his device.

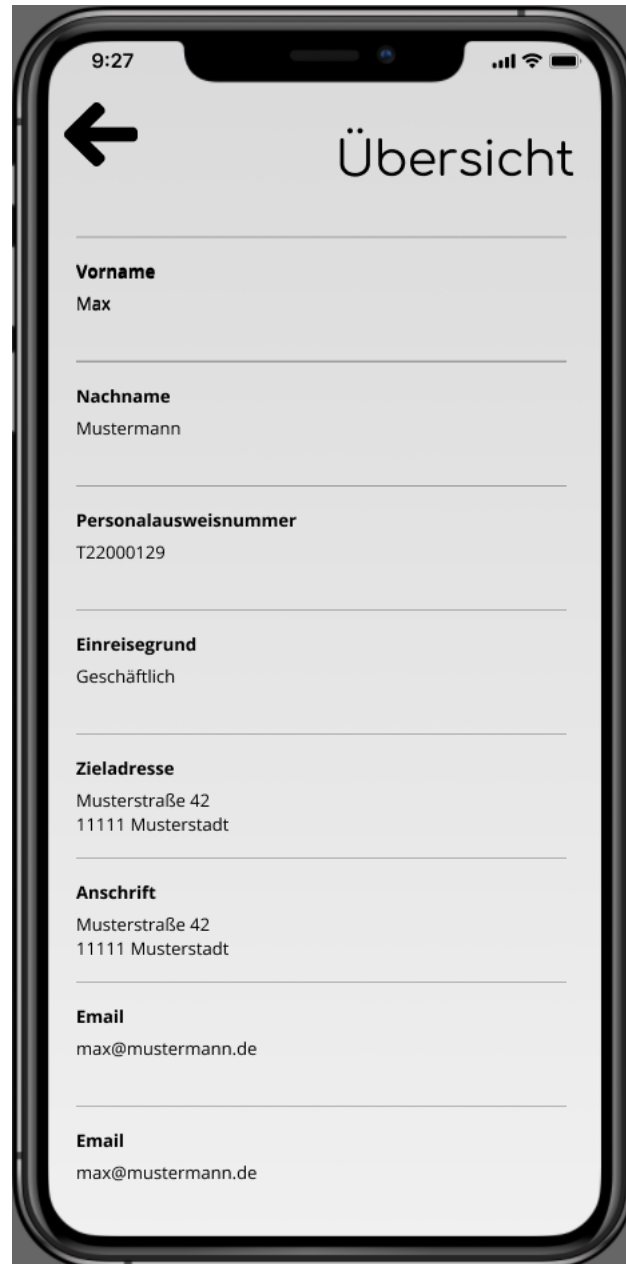


Fig. 3: Image of the Police Website

2.4 Backend: Connecting the Pieces

A flask webserver connects the individual parts of our application. A flask web server is used to serve the questionnaire to border crossers and answers the requests that are triggered when the border police scan's a QR code.

2.4.1 project.questionnaire module

Summary

Classes:

<i>LocalizedQuestion</i>	holds a localized question
<i>Question</i>	hold the description of a question
<i>Questionnaire</i>	holds all relevant data of a questionnaire

Reference

class project.questionnaire.**Questionnaire** (*global_id, language_map, questions*)

Bases: object

holds all relevant data of a questionnaire

global_questionnaire_id = ''

__init__ (*global_id, language_map, questions*)

Initialize self. See help(type(self)) for accurate signature.

language_map = {}

questions = []

initialization

:param str *global_id* – the globally unique identifier of this questionnaire. This is needed to connect questionnaires with answers. :param dict *language_map* – (dict of str: (str: str)) – contains the text-snippets used in this questionnaire in every supported language. The mapping is “language” -> “id” -> “text” :param list *questions* – (list of Question)the actual questions in the questionnaire

localized_questions (*language*)

class project.questionnaire.**Question** (*question_id, answer_type, options=[]*)

Bases: object

hold the description of a question

__init__ (*question_id, answer_type, options=[]*)

Initialize self. See help(type(self)) for accurate signature.

question_id = ''

answer_type = ''

options = []

initialization

:param str *question_id* – the identifier of the question. This is needed to identify the corresponding question test stored in the Questionnaire's *language_map* and associate questions with answers :param str *answer_type* – which kind of answer is expected (string, date, PLZ,...) :param list *options* – optional (array of strings) specifies all possible answers

```

class project.questionnaire.LocalizedQuestion (question, question_text, options_texts)
    Bases: project.questionnaire.Question
    holds a localized question
    __init__(question, question_text, options_texts)
        Initialize self. See help(type(self)) for accurate signature.
    question_text = ''
    options_texts = []
        initialization
        :param Question question – the question :param str question_text – the text of the question :param op-
        tions_texts – (list of str) the texts of the options

```

2.4.2 project.questions_from_json module

Summary

Functions:

<i>read</i>	Reads questionnaire from file
-------------	-------------------------------

Reference

```

project.questions_from_json.read(filename, context)
    Reads questionnaire from file
    :param filename – path to file containing questionnaire :rtype questionnaire.Questionnaire :return a questionnaire

```

2.4.3 project.flaskr package

Submodules:

project.flaskr.app module

project.flaskr.config module

Summary

Reference

Subpackages:

project.flaskr.routes package

Submodules:

project.flaskr.routes.overview module

Summary

Functions:

get_overview_blueprint

Reference

project.flaskr.routes.overview.**get_overview_blueprint** (*questions*)

project.flaskr.routes.questionnaire module

Summary

Functions:

get_questionnaire_blueprint

Reference

project.flaskr.routes.questionnaire.**get_questionnaire_blueprint** (*questions*)

project.flaskr.services package

Submodules:

project.flaskr.services.forms module

Summary

Reference

PYTHON MODULE INDEX

p

- `project.dict_generator`, 5
- `project.flaskr`, 11
 - `project.flaskr.config`, 11
 - `project.flaskr.routes`, 12
 - `project.flaskr.routes.overview`, 12
 - `project.flaskr.routes.questionnaire`, 12
 - `project.flaskr.services`, 12
 - `project.flaskr.services.forms`, 12
- `project.questionnaire`, 10
- `project.questions_from_json`, 11

Symbols

`__init__()` (*project.questionnaire.LocalizedQuestion method*), 11

`__init__()` (*project.questionnaire.Question method*), 10

`__init__()` (*project.questionnaire.Questionnaire method*), 10

A

`add_lang_dialog()` (*in module project.dict_generator*), 6

`answer_type` (*project.questionnaire.Question attribute*), 10

C

`cancel_inspect_mode()` (*in module project.dict_generator*), 6

`change_lang_inspect_mode()` (*in module project.dict_generator*), 6

`change_option()` (*in module project.dict_generator*), 6

`close_application()` (*in module project.dict_generator*), 8

`create_new_dict()` (*in module project.dict_generator*), 6

`create_table()` (*in module project.dict_generator*), 6

F

`fill_new_language()` (*in module project.dict_generator*), 6

`finish_new_language()` (*in module project.dict_generator*), 6

G

`get_overview_blueprint()` (*in module project.flaskr.routes.overview*), 12

`get_questionnaire_blueprint()` (*in module project.flaskr.routes.questionnaire*), 12

`global_questionnaire_id` (*project.questionnaire.Questionnaire attribute*), 10

H

`handle_table_click()` (*in module project.dict_generator*), 6

L

`language_map` (*project.questionnaire.Questionnaire attribute*), 10

`localized_questions()` (*project.questionnaire.Questionnaire method*), 10

`LocalizedQuestion` (*class in project.questionnaire*), 10

N

`new_dict_begin_dialog()` (*in module project.dict_generator*), 6

`new_dict_dialog()` (*in module project.dict_generator*), 6

`next_new_language()` (*in module project.dict_generator*), 6

O

`open_dialog()` (*in module project.dict_generator*), 6

`options` (*project.questionnaire.Question attribute*), 10

`options_texts` (*project.questionnaire.LocalizedQuestion attribute*), 11

P

`prev_new_language()` (*in module project.dict_generator*), 6

`project.dict_generator` (*module*), 5

`project.flaskr` (*module*), 11

`project.flaskr.config` (*module*), 11

`project.flaskr.routes` (*module*), 12

`project.flaskr.routes.overview` (*module*), 12

`project.flaskr.routes.questionnaire` (*module*), 12

`project.flaskr.services` (*module*), 12

`project.flaskr.services.forms` (*module*), 12

`project.questionnaire` (*module*), 10

`project.questions_from_json` (*module*), 11

Q

Question (class in *project.questionnaire*), 10
question_id (project.questionnaire.Question attribute), 10
question_text (project.questionnaire.LocalizedQuestion attribute), 11
Questionnaire (class in *project.questionnaire*), 10
questions (project.questionnaire.Questionnaire attribute), 10

R

read() (in module *project.questions_from_json*), 11

S

save_dialog() (in module *project.dict_generator*), 6
set_left_window() (in module *project.dict_generator*), 6

U

update_lang_inspect_table() (in module *project.dict_generator*), 6
updated_table() (in module *project.dict_generator*), 6

W

write_to_dict() (in module *project.dict_generator*), 8