ID Risk

Forms
(design/creation)

Rev 1.0

11022021

Goal

Goal

Goal

Goal

Direct input in a doc form

Advantages:

- WYSIWYG

- Printer friendly

Limitations:

- Embedding code in documents, whether PDF or DOC is neither safe nor reliable

- => Main validations only at the end

- => Semi-automatic filling only at the start or at the end

- Tablet/smartphone UX issues for some people

Goal

"Native"

Advantages:

- Quick data input
- Validations on the fly
- Usage of automatic actions while filling it

Limitations:

- Not printer friendly

So …

Flexibility is the keyword … so, we must find a way to use the best of the 2

So …

```
{
  "ver": "A001",
  "lang": "pt",
  "width": 550,
  "height": 750,
  "elements": [
    {
      "id": "label_01",
      "type": "label",
      "text": "Nome:",
      "x": 10,
      "y": 720
    },
    {
      "id": "label_02",
      "type": "label",
      "text": "Morada:",
      "x": 10,
      "y": 700
    },
    {
      "id": "text_01",
      "type": "textinput",
```

DATA DRIVEN FORMS

... two views

FORM VIEW

{
  "ver": "A001",
  "lang": "pt",
  "width": 900,
  "height": 1550,
  "page": "A4",
  "events_actions": [
    {"event": "OnFilled", "type": "qu
  ],
  "elements geom
  {

      type": "rectangle",

FORM
(json)

LIST VIEW

Inspector

Entity Description

ID

Name

Address

some value

Telefon
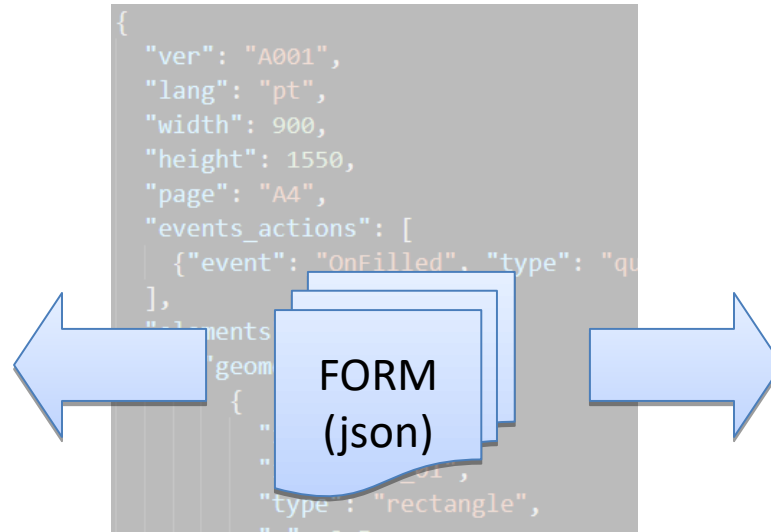
The user should be able to alternate between the two at any time. So, he chooses whatever he feels more comfortable

The *Form View* more adequate for PC or large Tablets and the *List View* for Normal/Small Tablets and Smartphones.

      "y": 3.3,
      "dx": 20.0,
      "dy": 5.0
    }
  ],

  "labels": [

FORM CREATION - Views

Forms are created in the *Form View*
(This view also corresponds to the print view)

The *List View* is automatically created
(the "designer" only needs to adjust the sections and the order of the fields)

FORM CREATION

- Three ways to build a form:
  - ➢ From scratch
  - ➢ Use the existing forms as raster images and add the *editables* on top
  - ➢ Importing: DOC → XML → JSON

FORM CREATION

- Directly from a database:

Uses the field metadata to automatically determine the type, length, max size and other sets and validations.

FORM CREATION

- Manually:

FORM CREATION

- Using raster Images to reuse existing paper forms (after all, why waisting existing work)

FORM CREATION

Name: Entity's name   Pho

```
{
  "zorder": 3,
  "page": 1,
  "section": "Entity Description",
  "id": "text_01",
  "type": "textinput",
  "placeholder": "Entity's name",
  "label": "label_01",
  "x": 100,
  "y": 150,
  "length": 20,
  "value": "",
  "db_field": "entity:name",
  "required": true
},
```

FORM CREATION

- *Sections* or *groups*, to organize how the form will appear in mobile (List View), with accordions, multi-screens, ….

FORM CREATION

- Setting sections and the order of the fields for the *List View*

Validations

- Client / Server reflection

- Example (Python/HTML5):

```
{
  "order": 3,
  "page": 1,
  "section": 2,
  "id": "text_01",
  "type": "email",
  "label": "label_01",
  "x": 2.5,
  "y": 1,
  "length": 20,
  "value": "",
  "db_field": "name",
  "required": true,
  "pattern": "[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,}$",
  "events_actions": [
    {"event": "onChange", "action": "uppercase"}
  ]
},
```

A Field	
Type:	Text
ID:	text_01
Name:	
Placeholder:	
Max length:	100
Required:	Yes
MultiLine:	No
DB Field:	
Pattern:	

Validations

```json
{
  "order": 3,
  "page": 1,
  "section": 2,
  "id": "text_01",
  "type": "email",
  "label": "label_01",
  "x": 2.5,
  "y": 1,
  "length": 20,
  "value": "",
  "db_field": "name",
  "required": true,
  "pattern": "[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,}$",
  "events_actions": [
    {"event": "onChange", "action": "uppercase"}
  ]
},
```

Client:

<input type="**email**" id="email" name="email" pattern="**[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$**" maxlength="**20**" **required**>

Validations

```
{
  "order": 3,
  "page": 1,
  "section": 2,
  "id": "text_01",
  "type": "email",
  "label": "label_01",
  "x": 2.5,
  "y": 1,
  "length": 20,
  "value": "",
  "db_field": "name",
  "required": true,
  "pattern": "[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,}$",
  "events_actions": [
    {"event": "onChange", "action": "uppercase"}
  ]
},
```

Server:

re.fullmatch("**[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$**", email)
not email
len(email) **<= 20**

Database and Storage

- Data sources:
  - Remote
  - Local
    - Forms ~ tables --- forms can have data tables associated it them, used for local queries, for populating options/lists and more.

Automatizations

- Mostly for auto-populate, enable/disable fields, etc
- 2 types:
  - Direct
  - Events / Actions
    - ➔ Ev. ex: OnFieldChanged, OnOpenForm, OnFormSubmit, ...

Automatizations
Direct

- Form field <-> database field
- 1:1
- No dependency between fields
- ***Direct automatic filling***
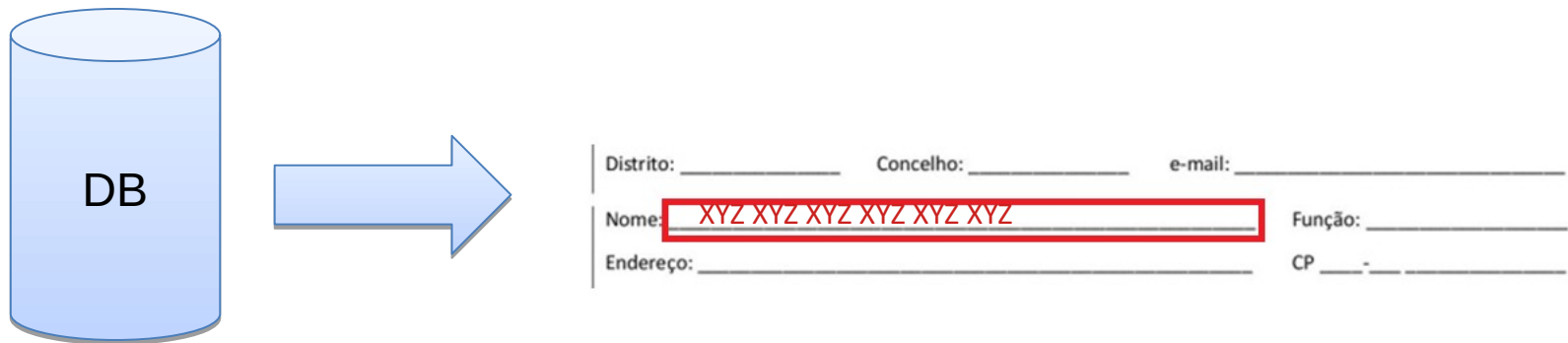
Automatizations
E/A

- Examples:

```json
{
  "order": 3,
  "page": 1,
  "section": 2,
  "id": "text_01",
  "type": "email",
  "label": "label_01",
  "x": 2.5,
  "y": 1,
  "length": 20,
  "value": "",
  "db_field": "name",
  "required": true,
  "pattern": "[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,}$",
  "events_actions": [
    {"event": "onChange", "action": "uppercase"}
  ]
},
```
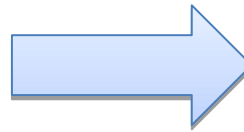
Max length:	100
Uppercase:	Yes
Required:	Yes
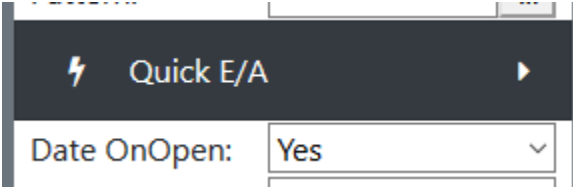
Uppercase all chars.

Automatizations
E/A

- Examples:

```
{
  "zorder": 8,
  "id": "date_01",
  "type": "date",
  "label": "label_08",
  "x": 3.0,
  "y": 9.0,
  "value": "",
  "db_field": "date",
  "format": "DD-MM-YYY",
  "required": true,
  "events_actions": [
    {"event": "onOpen", "action": "now"}
  ]
}
```

Quick E/A ▶

Date OnOpen: Yes ⌄

On opening the form and no value yet,
then automatically set the current date.

Data:

31/01/2021

Automatizations
E/A

- Examples:

```
{
  "id": "checkbox_2",
  "type": "checkbox",
  "label": "label_05",
  "checked": true,
  "x": 2.4,
  "y": 4.4,
  "events_actions": [
    {"event": "OnChecked", "target":["select_01"], "action": "disable"}
  ]
},
```

On checking this checkbox, disable field *"select_01"*

Automatizations
E/A

- Remote vs Local

```
"height": 1550,
"page": "A4",
"events_actions": [
  {
    "event": "OnFilled",
    "location": "remote",
    "type": "query",
    "fields":["name","address"],
    "action": "A005"
  },
```

When both the *name* and the *address* fields are filled, send request to server: *query A005*.

Automatizations
E/A

- Remote vs Local

```json
{
  "event": "OnFilled",
  "location": "local",
  "fields":["distrito"],
  "target": "concelho",
  "action": {
    "table":"nuts",
    "keys":["distrito"],
    "data":["concelho"],
    "operation": "all"
  }
}
```
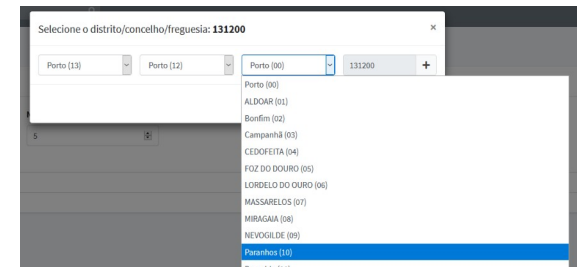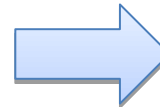
When the *distrito* field is filled, get all *concelhos* of that *distrito*, from local table *nuts*. Set the result in the *concelho* field.

Some Final Remarks

- So, the form creator requires:
  - *A form view editor*
  - *A list view*
  - *Events/Actions editor*
  - *Language mapping*
  - *Associated tools, like, a query and pattern editors, table creator/import/export, …*

Some Final Remarks

- Django
- HTML5/JS/TS
- IndexedDB
- PWA
- …

Technologies

- Signatures?
  - Digital signatures (ex: CC), just a normal drawing signature or none
- Mobile users/roles? Inspector and who else?