Advanced web methodology for flexible web development

Hiuram Antunes
ISEC/IPC, Instituto Superior de Engenharia, Instituto Politécnico de Coimbra
DEE, Dep. Engenharia Eletrotécnica
Coimbra, Portugal
hiuram.antunes@isec.pt

Inácio de Sousa Adelino da Fonseca
ISEC/IPC, Instituto Superior de Engenharia, Instituto Politécnico de Coimbra
DEE, Dep. Engenharia Eletrotécnica
Coimbra, Portugal
inacio@isec.pt

Abstract — **Web development is one of the ways to contribute to the digitization of organizations. However, being a relevant area, it has the contribution of countless companies and people worldwide. This high number of contributions necessarily leads to several trends and forms of development. This article explores one of these trends, namely the use of react on the frontend and nodejs on the backend. In terms of the data flow between the frontend and the backend, the GraphQL specification was used. The database is supported by Postgresql. The architecture and organization of development is crucial to find generic baseline sequences that support cross-cutting development between different information systems. This was the focus of the development in order to easily change the data structure and reflect this change in the frontend, without the need for maintenance in the code. This is a key step to allow the next stage, namely the development supported by advanced techniques based on artificial intelligence (this stage is under development and will be described in a next article by the authors).**

Keywords – Web Development; Javascript; Backend; Frontend.

I. INTRODUCTION

The development of applications has followed the tendency to move to the web side, to the detriment of the installation versions in the host operating systems (desktop versions). This trend can be seen, for example, with the existence of tools such as google docs (example: excel, word online), visual studio online [2], arduino online [1], etc.

Historically the development of web applications requires the knowledge of several technologies in the case of the frontend: at least html, css, javascript; and in the case of the backend: it depends on the technology used, with the trend being for javascript in nodejs. Obviously, the use of various software packages or frameworks also leads to an increase in the skill requirement by developers. Additionally, there is a tendency for most frameworks to support typescript (typed javascript), an extension provided by microsoft.

These aspects mean that a large part of the project development time is spent on the technical issues of the various technologies involved at the expense of specific applications. Not being a new issue, it is, however, an aspect present in application development for equipment, whether desktop, labtops, smartphones, or things - running windows, linux, macOS (OS X), Android, iOS, etc. Several commercial development tools try to solve this problem, for example, Microsoft's Visual Studio, or Idera's Rad Studio [3]. In the case of Rad Studio, the tool also includes an application specifically to design the graphic interface only by professionals in the design area, whose results can be incorporated into Rad Studio by the programmers.

II. STATE OF ART

In terms of development, there are several techniques, languages and frameworks involved. Realizing which path to follow can be complex as it is always necessary to consider the continuity of frameworks / languages in the medium to long term. In terms of data flow and its transfer between client and server, the most usual support is the http protocol, with the possibility of maintaining a permanent connection through sockets (websockets), in case there is a need for asynchronous server information. We can divide web development into several steps and tools:

- provision of data and its manipulation through the use of an API - usually on the backend. The REST API is one of the most common forms;
- development of the graphic aspect - usually by designers where the harmony, the colors used, the way to interact, and the sequence is extremely relevant and cared for. At this point, CSS allows you to decouple and work on this aspect - it may involve choosing UI libraries;
- choosing the specific framework for organizing the source code;
- tools for testing application functionality and code verification.

A. Data Provision

As of the writing of this article, the most relevant specifications, to get data from the server, are:

- Odata [4] (originally developed by Microsoft in 2007): is an OASIS standard REST API, used by Microsoft, SAP, CA, IBM and Salesforce;
- ORDS [5] (Oracle REST Data Services): Oracle Rest service API that in Oracle Database, Times Ten ans NoSQL products – it's an Oracle-centric API;

2021 16th Iberian Conference on Information Systems and Technologies (CISTI)
23 – 26 June 2021, Chaves, Portugal
ISBN: 978-989-54659-1-0

- GraphQL [6]: developed at Facebook, is now open-source. “GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools”;
- Falcor [7]: developed at Netflix. “Falcor lets you represent all your remote data sources as a single domain model via a virtual JSON graph. You code the same way no matter where the data is, whether in memory on the client or over the network on the server.”.

B. *Mockup and WireFrame*

The development of the prototype graphic aspect is fundamental, and some tools are: Figma, Adobe XD, Sketch, Pencil Project, InVision [8]. Figure 1 presents an example elaborated in Pencil.

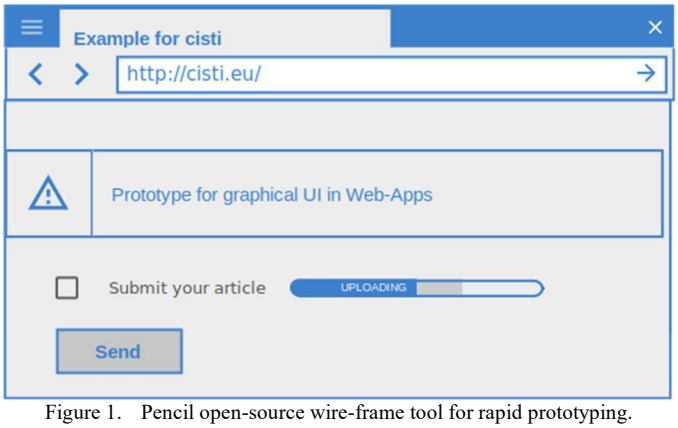

Figure 1. Pencil open-source wire-frame tool for rapid prototyping.

C. *UI Frameworks*

The graphic aspect can be supported by graphical frameworks, namely: Material UI, TailWind CSS, Chakra UI (css and javascript based ones) and css first frameworks that don't use javascript by default: bootstrap, materialize css, bulma [9] (see Figure 2).

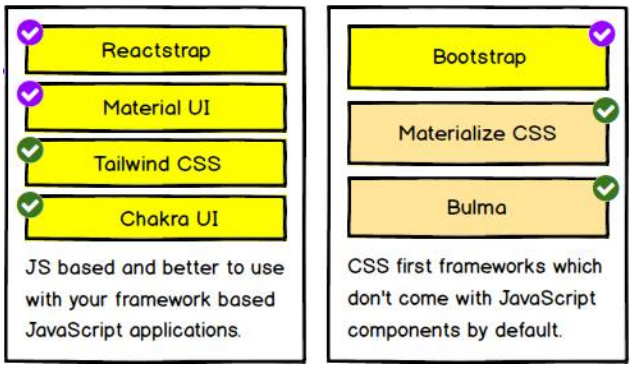

Figure 2. Developer Roadmap [10].

D. *Frontend frameworks*

In frontend development, the frameworks react, angular and vuejs should be highlighted. In this work, react was explored in terms of the frontend.

E. *Testing, testing, …, testing*

Software testing is important to verify functionality and requirements before the application is put into production. There are two trends lately:

- TDD - Test Driven Development: the requirements are converted into tests, even before the software development begins, forcing the programmer to better plan, thus leading to 100% test coverage of the entire development;
- BDD - Behavior Driven Development: thinking from the point of view of the expected behavior of a functionality by the user.

In addition to these trends are the normal tests that can be performed on an application:

- unit tests: testing individual modules;
- integration tests: check if several modules are working correctly when they start working together;
- functional tests: tests the functionalities from the user's point of view to verify that the behavior is as expected.

From a practical point of view, there are several frameworks to develop tests on web applications, namely: Mocha; Chai; Ava; Jasmine, Jest, Cypress; Enzyme; react-testing-library (Figure 3) [11].

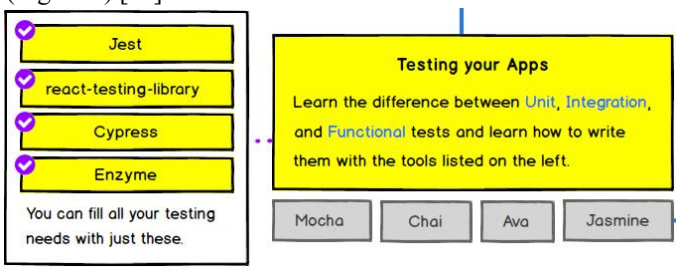

Figure 3. Testing tools [10].

III. DEVELOPMENT

To start the work, the most common configurations for web applications were studied, namely in terms of graphic organization of the items, reaching the conclusion of the next typical items (html div element with different formats).

Figure 4. Left: centered div inside a div. Right: two divs - menu and body.

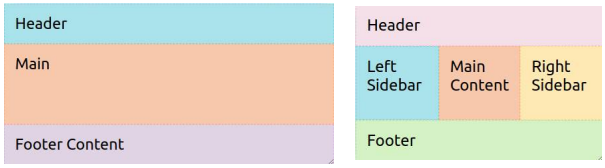

Figure 5. Basic containers.

ISBN: 978-989-54659-1-0

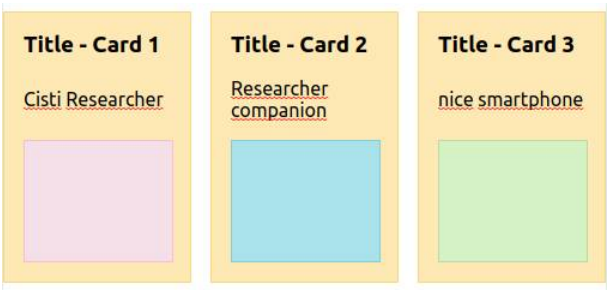

Figure 6. Basic containers.

Figures 4-6 show basic containers for html form elements.

The next step is to decide the data entry fields and ways to present the data, using native html components, or composite components. Figure 7 show two composite components and a native one.

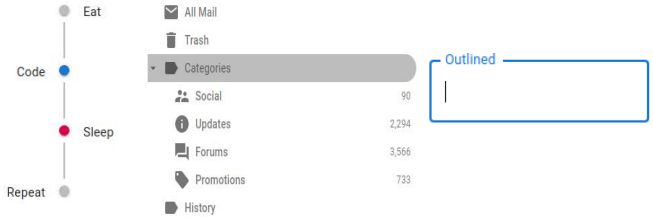

Figure 7. Display elements [12].

The next step is to define the correspondence between user information and components, making a correspondence between the type of information for each column / table in the database and the web components.

F. Backend

The backend uses GraphQL API. The main idea is to allow the user (designer) to define the accessible data, as well as the real data source from which they come. This aspect is crucial, as we can have a data source for authentication (database, ldap, ect) and an operational data source and another financial data source – as example. This method allows data integration easily from different manufactures – for example financial data can be at SAP system; users and access rules at Microsoft identity and access management; and specific app data at a local database.

This approach led to the choice of tools and the configuration of intermediate solutions to access data sources other than the native database and then to the construction of intermediate software that would easily allow changes in the type of data, in the number of fields requested, application of filters, checking access rules, etc.

Figure 8 shows the structure of the server that allowed studying the implementation as generic as possible to allow its reuse on a large scale.

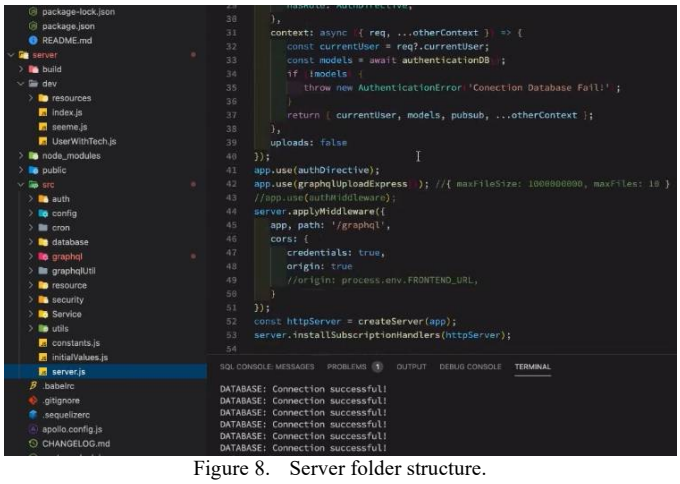

Figure 8. Server folder structure.

The server is using nodejs with express, jwt, apollo server, sequelize for local database access. Figure 9 shows the GraphQL types definition.

Figure 9. Example of GraphQL Data definition.

G. Frontend

In the frontend there is a need to create generic graphic components (in addition to the existing html natives) that can be used in the construction of the graphic visualization for the user, as well as in the correct sequence between screens. There is also a need to locally safeguard the status of each form in order to know at every moment what has been modified in order to make only real changes to the server, to minimize traffic and database operations. Traditionally, react already contains ways to manage the state of a component, or the state of components in general. Figure 10 presents some of the aspects to consider when developing react-based frontend applications.

ISBN: 978-989-54659-1-0

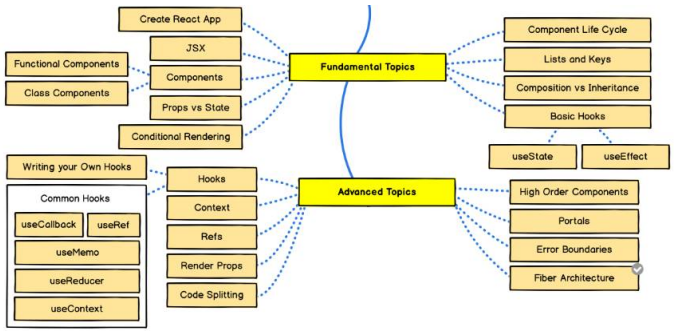

Figure 10. Part of the react ecosystem [10].

Taking these aspects into account, components were created in the frontend based on Material-UI, with the hypothesis of associating the type of data that they can contain at any given moment, or to put it another way, when "instantiating" a component to a form it is indicated also the type of data you should manage. The configurations of the various forms of the application can be built natively in react or arrive from the database in the form of configuration to define which aspect to render to the user.

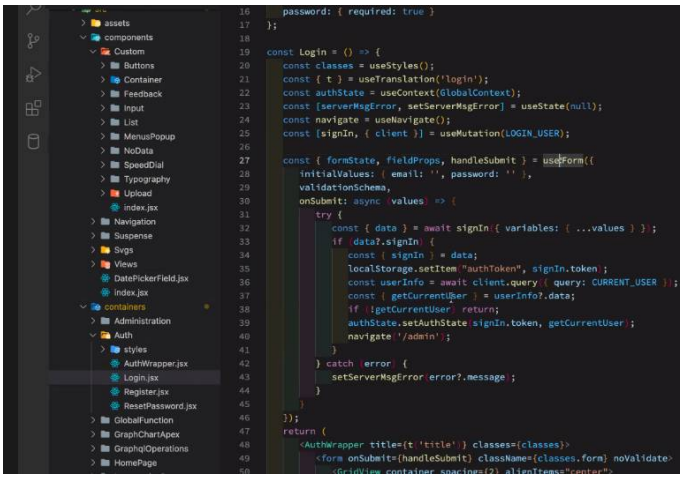

Figure 11. Part of the developed framework .

Figure 11 shows the created components and part of the login form, which is natively coded in react.

IV. Conclusions

This article presents a first robust study of ways of organizing code that are generic, as well as the code itself, with the aim of flexible and future automatic development depending on the type of data to be represented and / or collected. At the moment the organization obtained contains a set of lines of code that are used generically in any set of information that needs to be manipulated through web forms.

The future will go through studying artificial intelligence methodologies for automated writing of web applications in terms of this type of applications.

Acknowledgment

The authors acknowledge the contributions received from several friends.

References

[1] Amazon Arduino cloud, "Arduino online", online: https://create.arduino.cc/, last visit: 15-02-2021.

[2] Microsoft Visual Studio, "Visual Studio Online", online: https://visualstudio.microsoft.com/pt-br/services/github-codespaces/, last visit: 15-02-2021.

[3] Idera, Inc, "Rad Studio and FireMonkey Stencils", online: https://www.embarcadero.com/products/, last visit: 15-02-2021.

[4] Oasis, "Odata", online: https://www.odata.org, last visit: 15-02-2021.

[5] Oracle, "ORDS", online: https://www.oracle.com/database/technologies/appdev/rest.html, last visit: 15-02-2021.

[6] Facebook, "GraphQL", online: https://graphql.org/, last visit: 15-02-2021.

[7] Netflix, "Falcor", online: https://netflix.github.io/falcor/, last visit: 15-02-2021.

[8] Figma, Adobe XD, Sketch, Pencil Project, InVision, online: figma.com; adobe.com/products/xd.html; sketch.com/; invisionapp.com; last visit: 15-02-2021.

[9] CSS frameworks and javascript frameworks, online: materializecss.com; bulma.io; getbootstrap.com; chakra-ui.com; tailwindcss.com; material-ui.com; reactstrap.github.io/, last visit: 15-02-2021.

[10] Developer Roadmap, online: roadmap.sh/frontend, last visit: 15-02-2021

[11] Testing tools, online: mochajs.org; chaijs.com; github.com/avajs/ava; jasmine.github.io; jestjs.io; cypress.io; enzymejs.github.io/enzyme, last visit: 15-02-2021.

[12] W3C, "Aria AuthoringPractices", online: w3.org/TR/wai-aria-practices/#TreeView, las visit: 15-02-2021.

[13]

ISBN: 978-989-54659-1-0