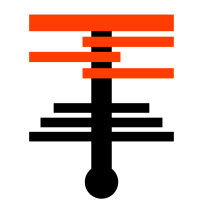


Project Submitted In the Context of the Course   
Info3301 **Software Engineering**



**Developed By  
Jad Mrad  
George Raed**

Lebanese University

Faculty of Sciences I

Department of computer Sciences

2017 - 2018

# Acknowledgements

Acknowledgements go here.

# Abstract

The chosen application/domain and the major steps done while achieving the project

# Table of Contents

[Acknowledgements 2](#_Toc534153053)

[Abstract 3](#_Toc534153054)

[Table of Contents 4](#_Toc534153055)

[Table of Figures 6](#_Toc534153056)

[Table of Tables 7](#_Toc534153057)

[Chapter I - Introduction 8](#_Toc534153058)

[I.1. The business domain 8](#_Toc534153059)

[I.2. About the modeled application 9](#_Toc534153060)

[2.1. Users Persona 9](#_Toc534153061)

[2.2. Analysis of the Existing Similar Programs 9](#_Toc534153062)

[I.3. Plan of the document 11](#_Toc534153063)

[Chapter II - Requirement Analysis and Specification 12](#_Toc534153064)

[II.1. Introduction 12](#_Toc534153065)

[II.2. Requirements Analysis 13](#_Toc534153066)

[2.1. Functional Requirements 13](#_Toc534153067)

[2.2. Non-functional Requirements 14](#_Toc534153068)

[II.3. Specification 14](#_Toc534153069)

[3.1. Use Case 14](#_Toc534153070)

[3.1. Use Cases Textual Description and Sequence Diagrams 15](#_Toc534153071)

[II.4. Addendums 18](#_Toc534153072)

[4.1. 18](#_Toc534153073)

[II.5. Conclusion 18](#_Toc534153074)

[Chapter III - Application Conception 18](#_Toc534153075)

[III.1. Introduction 18](#_Toc534153076)

[III.2. UML Class Diagram 18](#_Toc534153077)

[III.3. Sequence Diagrams 18](#_Toc534153078)

[III.4. Technical Design 19](#_Toc534153079)

[III.5. State Chart Diagram 19](#_Toc534153080)

[III.6. Maintenance and Delivery 19](#_Toc534153081)

[III.7. Conclusion 19](#_Toc534153082)

# Table of Figures

[Figure 1 Programming Language Market Shares - Early 2018 8](#_Toc534153083)

[Figure 2 - Oracle SceneBuilder 1.1 Early Release 9](#_Toc534153084)

[Figure 3 - Gluon SceneBuilder 8 11](file:///C:\Users\DELL\Documents\GitHub\EZFX\Documentation\EzFx%20Project%20Report.docx#_Toc534153085)

[Figure 4 - Use Case Diagram 14](file:///C:\Users\DELL\Documents\GitHub\EZFX\Documentation\EzFx%20Project%20Report.docx#_Toc534153086)

[Figure 5 - DFD of "Extract" Use Case 16](file:///C:\Users\DELL\Documents\GitHub\EZFX\Documentation\EzFx%20Project%20Report.docx#_Toc534153087)

# Table of Tables

[Table 1 Comparison of Existing Similar Programs & Frameworks 10](#_Toc534153088)

[Table 2 - Priority Description 15](#_Toc534153089)

[Table 3 - DTD of the “Import Project” Use Case 15](#_Toc534153090)

[Table 4 - DTD of the “Extract” Use Case 16](#_Toc534153091)

[Table 5 - DTD of the “Register” Use Case 17](#_Toc534153092)

1. Introduction
   1. The business domain

This project fits into the software development domain and is specifically directed towards Java programmers working on JavaFx or GUIs in general. We believe this is an ideal community to target as it is sizable with 3 Billion devices running java and its sizable market share at 22.9% as can be seen in figure 1. We also chose JavaFx as a platform to build upon as it is the most recent officially released framework for creating graphical user interfaces with updates coming frequently and as recent as 2 hours[[1]](#footnote-1) before this report was written.

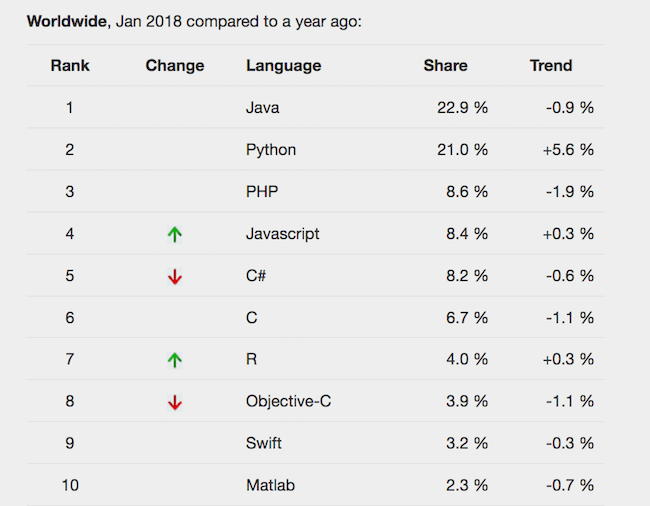


Figure 1 Programming Language Market Shares - Early 2018

In an ever-expanding community we believe to have stumbled upon a requirement that no one has provided yet based on our research. As will be discussed in detail below, any Java developer will in one way or another need our program be-it for large scale collaborations or a simple personal project.

* 1. About the modeled application

The program is intended to be used by anyone from beginners to experienced developers to design and implement JavaFx completely code-free. It can also be utilized by anyone who doesn’t know how to code which is useful on clients during the requirement gathering stage. With many more possibilities, the community needs such a code-free program not to mention the blueprints system that is so far unseen for JavaFx.

* + 1. Users Persona

Some key cases where our product would be optimal:

* Java developers creating GUI assets or prototypes with integrated logic
* Designers setting up a projects layout without requiring any application code
* Newcomers to JavaFx that wish to explore its potential
* Clients to show Software Engineers what they want
* Swing and/or AWS users switching over to JavaFx syntax
  + 1. Analysis of the Existing Similar Programs

There are quite a few programs and frameworks that could be considered alternatives or competitors to ours yet none of them carry the functionalities that EasyFx provides, mainly the handler management blueprints.

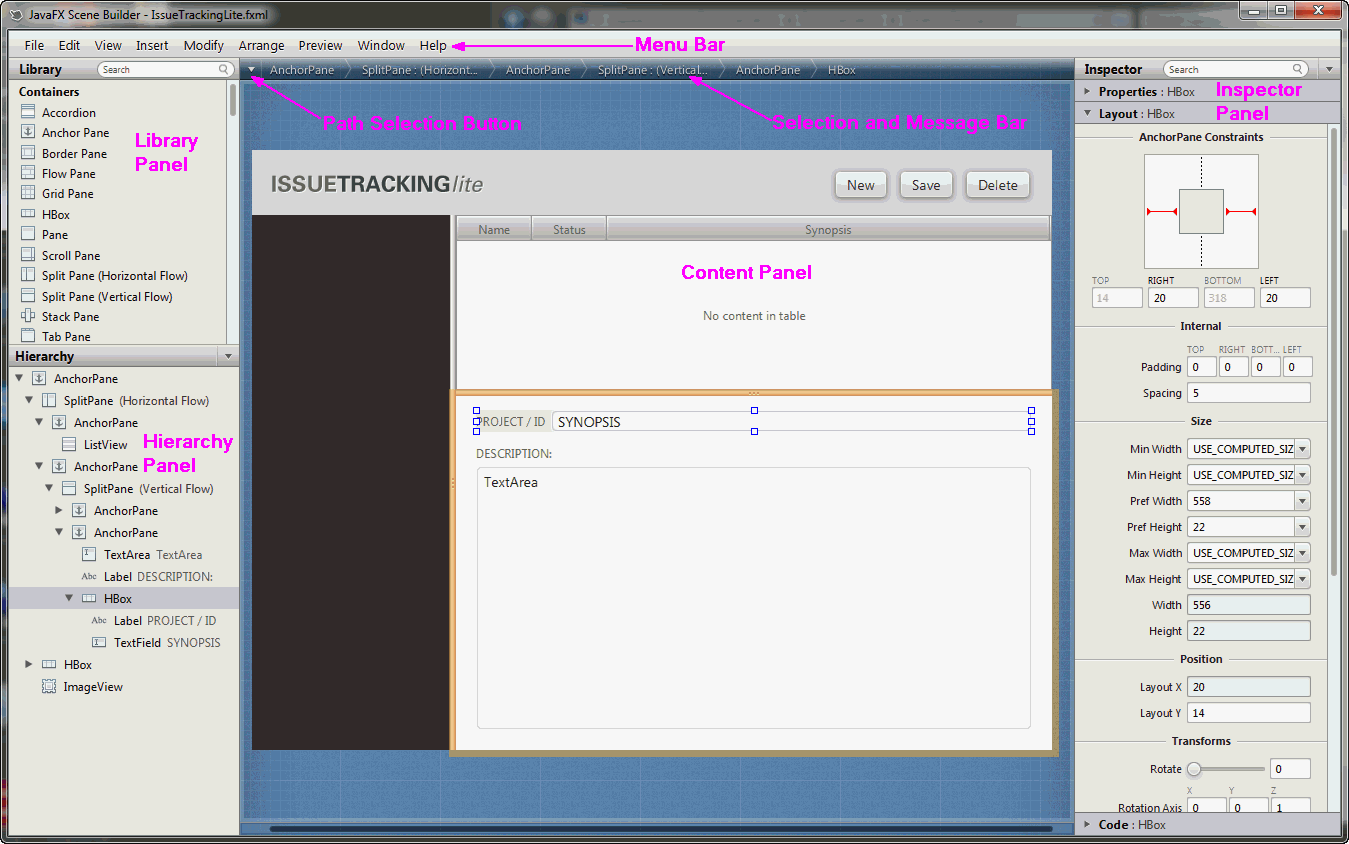


Figure 2 - Oracle SceneBuilder 1.1 Early Release

Oracle began SceneBuilder in late 2013 with the 2.1 release of JavaFX as a means of lightening the load of having to compile and execute code to view UI progress and check for any faults. It came at an early stage of Javas rise to dominate the programming world and thus was unopposed.

Though we view this as a gift to the community from Oracle and a major improvement that certainly made many projects easier, their decision to push their own Markup Language FXML, released in 2011, as the only possible output type is what prompted us to plan and go through with EasyFx.

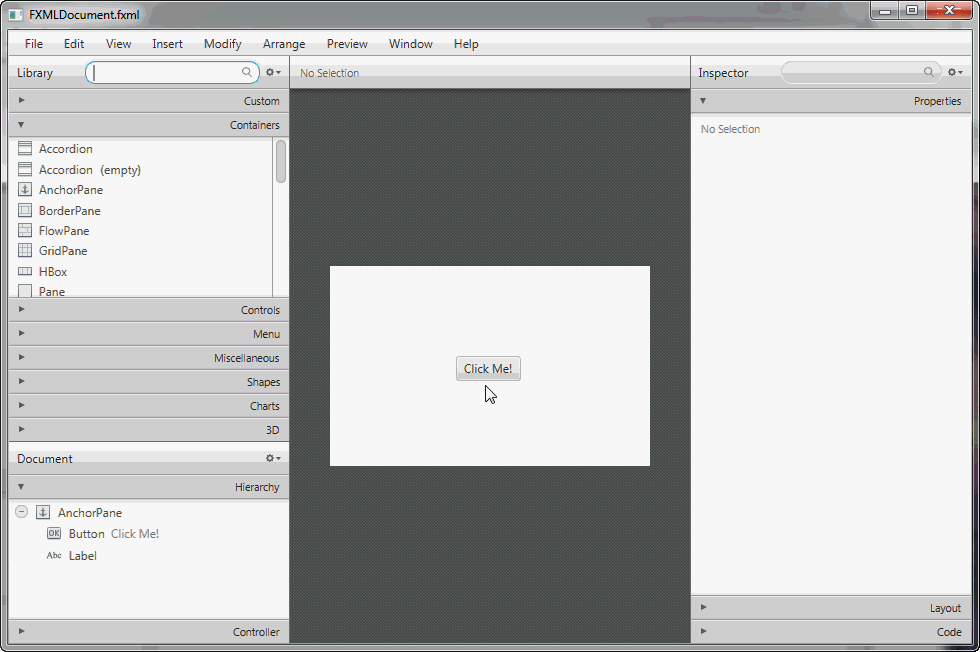
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Last Update | Up to Java version | Accepted Inputs | Output | Blueprints |
| Oracle Scene  Builder 1.1 | October 2013 | Java 7 | None | FXML | No |
| Oracle Scene Builder 2.0 | April 2014[[2]](#footnote-2) | Java 8 | FXML | FXML | No |
| Gluon SceneBuilder | June 2018 | Java 10 | FXML | FXML | No |
| EasyFx | January 2019 | Java 10 | EzML / FXML | JavaFx / EzML / FXML | Yes |

Table 1 Comparison of Existing Similar Programs & Frameworks

As is apparent in Table 1, our only proper competition is the Gluon continuation of Oracle’s SceneBuilder which took over post-2014 releases after Oracle halted development and favored a source-code only release. Though the developers over at Gluon have been doing a great job maintaining it, they continue to rely solely on FXML and have not provided a way for experienced developers to extract a proper JavaFx variation of their design.



Figure 3 - Gluon SceneBuilder 8



FXML simply takes away the extended libraries and functionalities that make JavaFx unique among its outdated rivals by replacing intricate Object Oriented syntax with a Markup Language. Overall what makes our program diverse is allowing the user a choice of output and integrating an essential part of development, event handling, in the form of blueprints.

* 1. Plan of the document

This document goes into detail on all stages of designing and maintaining this product from the basic requirements engineering process through development, prototyping, testing, and preparation for an official release. Future plans and mappings of projected evolution will also be included as well as extended versions or requirement gathering procedures. All these will be discussed in detail below.

1. Requirement Analysis and Specification
   1. Introduction

The following is a mildly modified condensation of the requirements collected through the initial meeting **[see addendum M1]** and follow-up meeting, phone calls, and questionnaire **[see addendums M2 through M4 and MQ]**. Unnecessary data has been omitted for the sake of simplicity. Note that the client has been simulated to better simulate the requirements gathering process. Our true client remains the Java developer community as a whole.

“EasyFx is a program that makes programming with JavaFx easier and code-free. Users should be able to drag and drop buttons, fields, and shapes onto the canvas. The user can edit the details of these items and the items can then be dragged around to change their position and can also be deleted. Selecting one of them sends the user to a blueprint-like page where he can manage what happens when interacting with each item (hovering, clicking, etc...).

Users can start on an empty canvas, use a pre-built template, or import their own projects that can be FXML or previous projects made on the program. Closing the program will first prompt the user to save their changes or cancel. Saving is done into a custom file type made for the program, as FXML, or a text file holding extracted JavaFx.

All this can be done using a guest account. Users can also register a local account with an email address. This will allow them to save their progress directly onto the local account instead of a file for ease of use. They are prompted upon creating an account to sign up for the newsletter to receive information about future releases. Registered users can access a settings tab to manage their account details, logout, clear data, or sign up/out of the newsletter. They can also switch between a light and dark theme.

The system should ensure that user data, encrypted locally, can only be accessed by the user himself and no one else. Passwords should never be viewable at the point of entry or any other time. Users shall receive notification of any profile change or unauthorized login via email.

Use a custom file extension to save created projects. The program itself shouldn’t be very big and should fit in a portable, executable JAR file.”

* 1. Requirements Analysis

The following is the extraction of functional and non-functional requirements from the passage.

“Custom file type made for the program” has been replaced with “EZML”, our program’s own markup language that constructs the UI.

* + 1. Functional Requirements

Guests can:

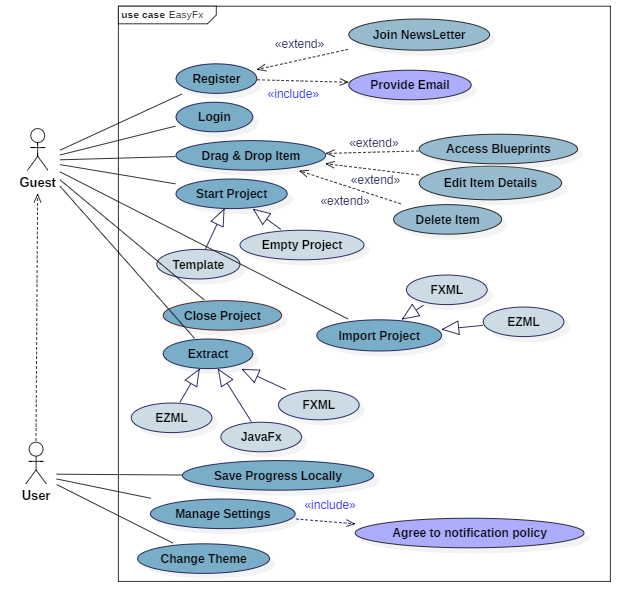
* Register a local account with an email address. **Feasible**
* Drag and drop buttons, fields, and shapes onto the canvas. **Feasible**
* Edit the details of these items. **Feasible**
* Items can then be dragged around to change their position. **Not Feasible [see addendum M3]**
* Delete items. **Feasible**
* Manage events with blueprints functionality. **Feasible**
* Start an empty project. **Feasible**
* Start a template project. **Feasible**
* Import EZML or FXML project. **Feasible**
* Close the program. [Customized closing sequence] **Feasible**
* Save/Extract as EZML, FXML, or JavaFx. **Feasible**

Registered users can:

* Save their progress directly onto the local account. **Feasible**
* When creating an account sign up for the newsletter to receive information about future releases. **Feasible**
* Access a settings tab to manage their account details, logout, clear data, or sign up/out of the newsletter. **Feasible**
* Receive notifications of any profile change or unauthorized login via email. **Not Feasible [see addendum M4]**
* Change the products theme. (light/dark) **Feasible**
  + 1. Non-functional Requirements

The system should:

* Allow users to drag & drop items. **Usability**
* Ensure that user data, encrypted locally, can only be accessed by the user himself and no one else. **Confidentiality**
* Passwords should never be viewable at the point of entry or any other time. **Access Security**
* The program itself shouldn’t be very big and should fit in a JAR file. **Portability**



* Run without using high maintenance servers **Development Requirement**
  1. Specification
     1. Use Case

Figure 4 - Use Case Diagram

|  |  |  |
| --- | --- | --- |
| Priority |  | Details |
| 1 | Cosmetic | - Serves no purpose |
| 2 | Add On | - Extension that runs alongside main functionalities |
| 3 | Neutral | - Secondary functions |
| 4 | Semi Essential | - Required for proper functionality. Failure won’t cause error |
| 5 | Essential | - Failure will most probably cause error state |

* + 1. Use Cases Textual Description and Sequence Diagrams

Table 2 - Priority Description

* + - 1. “Import Project” Use Case:

|  |  |  |
| --- | --- | --- |
| **Number** | #UC\_DTD\_001 | |
| **Name** | Import Project | |
| **Summary** | Extract EZML project from input | |
| **Priority** | 4 | |
| **Post-conditions** | Project is constructed and opened | |
| **Primary Actor** | User | |
| **Trigger** | Import option from File menu | |
| **Main Scenario** | **Step** | **Action** |
|  | 1 | User browses for target file |
|  | 2 | System validates file extension |
|  | 3 | System constructs UI from project |
| **Extensions** | **Step** | **Branching Action** |
|  | 2a | File extension is invalid |
|  | 2b | System notifies user: “Invalid File Type” |
|  | 2c | Import cancelled |
|  | 3a | System runs into error extracting data |
|  | 3b | User is notified of the error |
|  | 3c | Import cancelled |

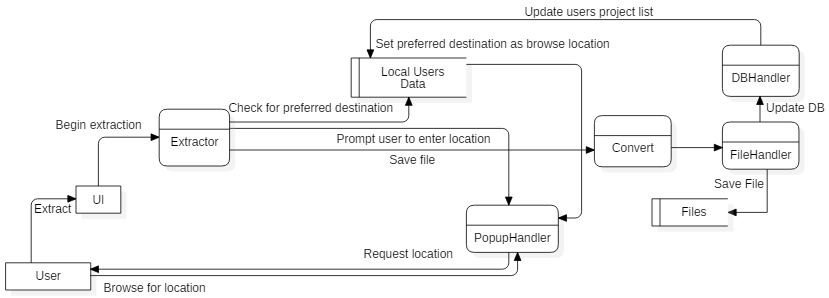
Table 3 - DTD of the “Import Project” Use Case

* + - 1. “Extract” Use Case

|  |  |  |
| --- | --- | --- |
| **Number** | #UC\_DTD\_002 | |
| **Name** | Extract | |
| **Summary** | Export created project into a specific format | |
| **Priority** | 4 | |
| **Preconditions** | **N/A** | |
| **Post-conditions** | User now has a file in a specific format of his creation | |
| **Primary Actor** | User | |
| **Trigger** | Export option from File menu | |
| **Main Scenario** | **Step** | **Action** |
|  | 1 | User selects save location (Browse) |
|  | 2 | User selects format (EZML, FXML, or raw JavaFx) |
|  | 3 | User enters file name |
|  | 4 | System checks if a file of that name/extension exists in target |
|  | 5 | User provides preferred save location |
| **Extensions** | **Step** | **Branching Action** |
|  | 3a | File with same name exists, prompt user to over-write file |
|  | 3b1 | User agrees. File is over-written |
|  | 3b2 | User denies. Cancel extraction |

Table 4 - DTD of the “Extract” Use Case

Figure 5 - DFD of "Extract" Use Case



* + - 1. “Register” Use Case:

|  |  |  |
| --- | --- | --- |
| **Number** | #UC\_DTD\_001 | |
| **Name** | Register | |
| **Summary** | Register local account and input details | |
| **Priority** | 2 | |
| **Post-conditions** | User has an active local account in the database | |
| **Primary Actor** | User | |
| **Trigger** | Register button from Login scene | |
| **Main Scenario** | **Step** | **Action** |
|  | 1 | User provides name and email |
|  | 2 | Email is verified (?) |
|  | 3 | User provides a valid password |
|  | 4 | System checks password strength |
|  | 5 | User provides preferred save location |
|  | 6 | System prompts user to join newsletter |
|  | 7 | User accepts |
|  | 8 | User added to news list |
| **Extensions** | **Step** | **Branching Action** |
|  | 2a | System notifies user: “Email is not valid” |
|  | 2b | User prompted to retry step 1 |
|  | 3a1 | System notifies user: “Password is weak” |
|  | 3b1 | Passwords do not match |
|  | 3a2/3b2 | User sent back to step 3 |
|  | 6a | User denies |
|  | 6b | User set to be notified about newsletter on next login |
| **Open Issues** | I001\_1 | Email verification system not confirmed functional yet |

Table 5 - DTD of the “Register” Use Case

* 1. Addendums
     1. Meetings:
        1. M1: Initial conception meeting
        2. M2: Follow-up in depth meeting
        3. M3: Phone call regarding feasibility
        4. M4: Final pre-release meeting
     2. Questionnaire:
        1. MQ: JavaFx developers
  2. Conclusion

This chapter

1. Application Conception
   1. Introduction

In the previous chapters, we have talked about the features that should be offered by our application... This reveals that the following entities are implied in the process… In this chapter we give, using UML class diagrams, the conceptual model that clarifies these entities, their underlying handled data, roles in the application and associations with each other.

* 1. UML Class Diagram

The whole picture can be cut into several smaller diagrams, each focusing on a specific set of interrelated entities and concepts.

Every diagram should be well annotated and explained in the sequel. Every choice made (association, cardinality, abstraction, etc.) and every used symbol should be justified and well explained.

* 1. Sequence Diagrams

Can be given for the main scenarios to show the sequence of functions calls between objects and instances of the various proposed classes.

* 1. Technical Design

About pieces of code, database choice, web techniques, languages choice…

* 1. State Chart Diagram

Opt for state chart diagrams only if one (or more) of your key and central entities change state as the process progresses and when different use cases are executed, and you want to capture this crucial business logic.

* 1. Maintenance and Delivery

This part is also optional and it is to be done only if you find that you have enough concrete elements in order to realize it.

If you ever want to get started, then give an estimated delivery date; A plan of successive deliveries in case you have several batches.

Also give your plan to support your client by helping him adopt your project, put it into service and then your strategy to help him and keep his service in perfect condition in the future.

* 1. Conclusion

This chapter presented our application design. First, we introduced…

After that, we presented...

In the next chapter, we make a complete feasibility study and draw out the plan of our project realization, risk management and client support and satisfaction strategy...

1. [JavaFx GitHub Release](github.com/javafxports) [↑](#footnote-ref-1)
2. [Oracle SB 2.0 Release notes](docs.oracle.com/javase/8/scene-builder-2/release-notes) [↑](#footnote-ref-2)