

University of Brasília - UnB
Faculty Gama - FGA
Bachelor of Software Engineering

Impact of a Pseudocode Online Judge on Programming Language Learning

Author: Geovana Ramos Sousa Silva
Supervisor: Giovanni Almeida Santos, M.Sc.

Brasília, DF
2021



Geovana Ramos Sousa Silva

Impact of a Pseudocode Online Judge on Programming Language Learning

In partial fulfillment of the requirements for
the degree of Bachelor of Software Engineer-
ing.

University of Brasília - UnB

Faculty Gama - FGA

Supervisor: Giovanni Almeida Santos, M.Sc.

Brasília, DF

2021

Geovana Ramos Sousa Silva

Impact of a Pseudocode Online Judge on Programming Language Learning/
Geovana Ramos Sousa Silva. – Brasília, DF, 2021-
126 p. : il. color.) ; 30 cm.

Supervisor: Giovanni Almeida Santos, M.Sc.

Undergraduate Thesis – University of Brasília - UnB
Faculty Gama - FGA , 2021.

1. calango. 2. online judge. I. Giovanni Almeida Santos, M.Sc.. II. University
of Brasília. III. Faculty UnB Gama. IV. Impact of a Pseudocode Online Judge on
Programming Language Learning

CDU 02:141:005.6

Geovana Ramos Sousa Silva

Impact of a Pseudocode Online Judge on Programming Language Learning

In partial fulfillment of the requirements for
the degree of Bachelor of Software Engineer-
ing.

Approved. Brasília, DF, October 14, 2021:

Giovanni Almeida Santos, M.Sc.
Supervisor

Edna Dias Canedo, Ph.D
Examiner

Milene Serrano, Ph.D
Examiner

Brasília, DF
2021

Acknowledgements

First and foremost, I thank God for providing me with the opportunity to step into the excellent world of science.

From the bottom of my heart, I am extremely grateful to my parents, Luiz and Mariana, for their love, unconditional support, encouragement, for giving me strength when I had none left, and for being the first to believe in whatever I have ever set as a goal to my life.

I would also like to express my gratitude and appreciation for Prof. Giovanni Santos for giving me the opportunities to grow as a professional and for the research partnership during all of these years. I have benefited greatly from your knowledge and academic guidance.

I cannot forget to thank my course colleagues that shared great experiences with me and the ones that participated in this work. Special thanks to my dearest and best friends that provided me with the best years of my life: Gabriela Medeiros, Gabriel Martins, Guilherme Guy, Joberth Rogers, and Daniel Maike.

Last but not least, I thank the Faculty Gama professors, the University of Brasília, the Laboratory for Decision Making Technologies (LATITUDE), and everyone that has been part of my academic journey until now.

*“Entrust your works to the LORD,
and your plans will succeed.
(Proverbs 16:3)*

Resumo

O Calango é uma linguagem de programação criada para ensinar programação de computadores a iniciantes. Ele foi desenvolvido partindo do pressuposto de que falantes do português que não dominam a língua inglesa têm dificuldade em aprender linguagens de programação que possuem palavras-chave em inglês. Por isso, as palavras-chave do Calango estão mais próximas do português falado e é uma linguagem de programação simplificada. No entanto, o Calango não possui o suporte de uma ferramenta valiosa para o ensino de linguagens de programação: um juiz online. Os juízes online permitem que os professores testem, de forma automática e rápida, o código dos seus alunos, ao passo que os alunos se beneficiam de uma resposta instantânea, tendo a oportunidade de corrigir seu código e aprender com seus erros. Portanto, este trabalho aborda o desenvolvimento do Calango Online Judge (COJ) e sua experimentação em aulas que ensinam introdução à programação na Universidade de Brasília (UnB). O seu desenvolvimento e experimentação foram executados em ciclos paralelos de Pesquisa-Ação e Desenvolvimento Rápido de Aplicações, o que desencadeou melhorias no software. Tanto a metodologia de ensino-aprendizagem adotada quanto o COJ foram avaliados através de três pesquisas aplicadas em períodos diferentes de cada ciclo. Padrões comportamentais dos estudantes foram identificados por meio da análise dos dados do COJ.

Palavras-chaves: juiz online. Calango. introdução à programação.

Abstract

Calango is a programming language created for introducing computer programming to novices. It was developed based on the assumption that Portuguese speakers that do not master the English language have difficulties learning programming languages with English keywords. Hence, Calango keywords are closer to spoken Portuguese, and it is a simplified programming language. However, Calango lacks the support of a valuable tool for teaching programming languages: an online judge. Online judges allow professors to automatically and quickly test their students' code, whereas students benefit from instant feedback, having the opportunity to correct their code and learn from their mistakes. Therefore, this work approaches the development of the Calango Online Judge (COJ) and its experimentation in classes that teach introductory programming at the University of Brasília (UnB). The development and experimentation were run in parallel cycles of Action Research and Rapid Application Development (RAD), which triggered improvements in the software. Both the teaching-learning methodology and COJ were evaluated through three surveys applied at different periods in each cycle. Student behavioral patterns were identified by analyzing COJ's data.

Keywords: online judge. Calango. introductory programming.

List of Figures

Figure 1 – Calango IDE	31
Figure 2 – Research Methodology	39
Figure 3 – RAD Methodology	42
Figure 4 – Activities timeline	45
Figure 5 – COJ Architecture	50
Figure 6 – Login page	54
Figure 7 – Home page (professor’s view)	55
Figure 8 – Scheduled lists page (professor’s view)	56
Figure 9 – Schedule’s detail page (professor’s view)	56
Figure 10 – Schedule’s results page	57
Figure 11 – Problems page	58
Figure 12 – Problem’s details page	58
Figure 13 – Submissions page	59
Figure 14 – Classes page	60
Figure 15 – Help page	61
Figure 16 – Deployment pipeline	62
Figure 17 – Number of accepted and unaccepted submissions per list	72
Figure 18 – Distribution of errors of unaccepted submissions per list	72
Figure 19 – Average number of unaccepted submissions per student on each ques- tion of a list.	73
Figure 20 – Students’ submissions per day of list grouped by problems solved . . .	75
Figure 21 – Number of students per class grouped by list conclusion, gender, and semester	76
Figure 22 – Entity-relationship diagram of the Django application	91
Figure 23 – Logical database schema generated by the ORM for PostgreSQL	92
Figure 24 – Sequence diagram showing the interaction between the judge service and the Calango Interpreter to judge a submission from the Web App .	93
Figure 25 – (S1Q1) What degree do you pursue?	101
Figure 26 – (S1Q2) How important do you think programming knowledge is to the degree you have chosen (or intend to choose) and to your professional life?	102
Figure 27 – (S1Q3) What semester of your degree are you in?	102
Figure 28 – (S1Q4) Have you ever been in contact with programming languages? .	103
Figure 29 – (S1Q5) If you have been in contact with programming languages, mark the languages with which you had contact.	103
Figure 30 – (S1Q6) Have you ever heard of the Calango language?	104

Figure 31 – (S1Q7) In relation to Calango and your expectations regarding the course, you...	104
Figure 32 – (S1Q8) Do you know what an online judge is?	105
Figure 33 – (S1Q9) In relation to online judges and your expectations regarding the course, you...	105
Figure 34 – (S2Q2) Regarding the use of CALANGO as a tool and programming language, in general, you were...	106
Figure 35 – (S2Q4) Regarding the use of COJ as an online judge, in general, you were...	106
Figure 36 – (S2Q5) What did you think about COJ's judgment?	107
Figure 37 – (S2Q6) In your opinion, would it be interesting to have more questions in COJ, in addition to the lists, for practicing?	107
Figure 38 – (S2Q7) In your opinion, would it be interesting for the COJ to approach theoretical content, such as multiple-choice questions?	108
Figure 39 – (S2Q8) What did you think about the level of COJ problems?	108
Figure 40 – (S2Q9) What did you think about the quality of COJ problems and their test cases?	109
Figure 41 – (S2Q10) What did you think about the number of questions on the lists?	109
Figure 42 – (S2Q11) Did the submission chart on COJ's home page encourage you to make better submissions to improve your results?	110
Figure 43 – (S3Q1) Did you think Calango made it easier for you to learn programming logic?	113
Figure 44 – (S3Q2) Did you think Calango made the transition to the C language easier?	113
Figure 45 – (S3Q3) What did you think about Calango's usage time?	114
Figure 46 – (S3Q4) For you, did the fact that Calango is in Portuguese facilitate your learning?	114
Figure 47 – (S3Q5) What is your level of knowledge in English?	115
Figure 48 – (S2Q6) Did you find that the online judges used strengthened your practical knowledge in programming?	115
Figure 49 – (S3Q7) For you, which methodology makes you more motivated in the course?	116
Figure 50 – (S3Q8) Do you think online judges (COJ and URI) have made the quality of your codes better than if you had to deliver directly to the teacher without prior feedback?	116
Figure 51 – (S3Q9) What did you think about taking the course in this class?	117
Figure 52 – (S3Q10) In your opinion, were all the contents of the course satisfactorily addressed?	117

Figure 53 – (S3Q11) Most of the time in this course, did you feel motivated or unmotivated?	118
Figure 54 – (S3Q12) What did you think about the pace of the course?	118
Figure 55 – (S3Q13)(2020.2-14A) What did you think about the workload of the course?	119
Figure 56 – (S3Q13)(2021.1-CC) What did you think about the workload of the course?	119
Figure 57 – Original SUS questionnaire	125

List of Tables

Table 1 – Corresponding Calango keywords for C keywords.	32
Table 2 – COJ’s functional requirements	48
Table 3 – COJ’s non-functional requirements	49
Table 4 – Results returned by the Judge Microservice	53
Table 5 – Amount of responses to the surveys	65
Table 6 – Percentage of students that selected positive impressions about COJ’s attributes in survey questions.	69
Table 7 – SUS scores obtained from survey	69
Table 8 – General students results from COJ	71
Table 9 – Deadlines of each list applied during the experiment	74
Table 10 – Students’ average attempts until acceptance per URI problem	75
Table 11 – Average attempts per student for each concluded EP grouped by list conclusion, gender, and semester	76
Table 14 – (S2Q12)(2021.1(DD)) If you want to add something, use the field below to commend or suggest improvements for Calango or COJ.	110
Table 12 – (S2Q12)(2020.2-14A) If you want to add something, use the field below to commend or suggest improvements for Calango or COJ.	111
Table 13 – (S2Q12)(2021.1-CC) If you want to add something, use the field below to commend or suggest improvements for Calango or COJ.	112
Table 15 – (S3Q14)(2020.2-14A) If you want to add something, use the field below to commend or suggest improvements to the course.	120
Table 16 – (S3Q14)(2021.1-CC) If you want to add something, use the field below to commend or suggest improvements to the course.	121
Table 17 – (S3Q14)(2021.1-DD) If you want to add something, use the field below to commend or suggest improvements to the course.	122
Table 18 – Validated European Portuguese version of the SUS questionnaire	126

List of abbreviations and acronyms

ANTLR	ANother Tool for Language Recognition
COJ	Calango Online Judge
EBNF	Extended Backus–Naur Form
EP	Evaluative Problem
IaaS	Infraestructure as a Service
IDE	Integrated Development Environment
MVP	Minimum Viable Product
NEP	Non-Evaluative Problem
ORM	Object-Relational Mapper
RAD	Rapid Application Development
RDBMS	Relational Database Management System
SSH	Secure Shell
SUS	System Usability Scale
UnB	University of Brasília

Contents

1	INTRODUCTION	25
1.1	Contextualization	25
1.2	Justification	26
1.3	Research Question	26
1.4	Aims and Objectives	27
1.5	Manuscript Organization	27
2	BACKGROUND AND THEORY	29
2.1	Meaningful Learning in Introductory Programming	29
2.2	Calango	30
2.2.1	Similar Tools	31
2.3	Online Judge	33
2.3.1	Definition	33
2.3.2	History	34
2.3.3	Advantages of Automated Assessment	35
2.3.4	Important Features	36
2.4	Chapter Summary	37
3	METHODOLOGY	39
3.1	Research Methodology	39
3.1.1	Action Research	40
3.1.2	Surveys	40
3.1.3	Data Analytics	41
3.1.4	Research Tools	41
3.2	Software Development Methodology	41
3.2.1	Rapid Application Development (RAD)	41
3.2.1.1	Requirements Planning	42
3.2.1.2	Prototype Cycles	43
3.2.1.3	Cutover Phase	43
3.2.2	Development Tools	44
3.3	Activities Timeline	44
3.4	Chapter Summary	44
4	CALANGO ONLINE JUDGE	47
4.1	Requirements	47
4.1.1	Functional Requirements	47

4.1.2	Non-functional Requirements	47
4.2	Software Architecture	50
4.2.1	Judge Microservice	51
4.2.2	Web Application	52
4.3	User Interfaces	54
4.3.1	Authentication	54
4.3.2	Home	54
4.3.3	Scheduled Lists	55
4.3.4	List Results	57
4.3.5	Problems	57
4.3.6	Submissions	58
4.3.7	Classes	59
4.3.8	Help	60
4.4	DevOps	61
4.4.1	Pipeline	61
4.4.2	Infrastructure	62
4.5	Chapter Summary	62
5	RESULTS AND DISCUSSION	65
5.1	Surveys	65
5.2	First Survey Results	66
5.3	Second Survey	67
5.4	Third Survey Results	70
5.5	Data Analytics	71
5.6	Threats to Validity	77
5.7	Chapter Summary	77
6	CONCLUSION	79
	REFERENCES	81
	APPENDIX	87
	APPENDIX A – SOURCE CODE REPOSITORIES	89
	APPENDIX B – ARTIFACTS	91
	APPENDIX C – DATA ANALYTICS SQL	95
	APPENDIX D – SURVEYS	101

D.1	First Survey Results	101
D.2	Second Survey Results	106
D.3	Third Survey Results	113
	 ANNEX	 123
	ANNEX A – SUS QUESTIONNAIRE	125
A.1	Original SUS	125
A.2	European Portuguese SUS	126

1 Introduction

This chapter details the problem that motivated this work, defines the proposed solution, sets the objectives, and describes how this document is organized.

1.1 Contextualization

As educational institutions go deeper into STEM (Science, Technology, Engineering, & Mathematics) education, programming becomes a relevant subject in areas beyond technology. Writing efficient algorithms requires more abilities than knowing a programming language, such as problem-solving, mathematical skills, abstraction, critical thinking, creativity, time management, and English ([MEDEIROS; RAMALHO; FALCÃO, 2019](#)).

Learning your first programming language is not always an easygoing experience, as shown by the high failure rates in introductory programming courses worldwide ([WATSON; LI, 2014](#)). Several social, economic, and educational factors can be associated with this problem when looking at students as a whole. If we look deeper into each country, it is possible to find specific barriers to novice learners. In [Canedo, Santos and Freitas \(2017\)](#), 66% of interviewed Brazilian students considered introductory programming courses as one of the most challenging courses, and 21% considered it very challenging.

Apart from the same difficulties every novice face when learning programming languages, non-English speakers also have the language obstacle, because most of the programming languages have English-based keywords and commands ([VEERASAMY; SHILLABEER, 2014](#)). Starting an introductory programming course with professional languages, such as C, may hinder students from developing logical thinking and problem-solving abilities. Because of the complex syntax and foreign keywords, they spend more time trying to figure out how to program in that language instead of focusing on the problem.

Therefore, some professors start courses with pseudocode. Since pseudocode is a generic language, learners can concentrate on the problem being solved instead of worrying about syntax and semantics, and it is possible to write algorithms in their native language. For helping specifically Portuguese-speaker learners, there is a pseudo-language called Portugol. However, pseudocode is not executable and students cannot test their solutions. Therefore some tools were created to execute Portugol.

One of the initial implementations of Portugol was a symbiosis of the Portuguese language and the programming languages: Pascal and Algol. The most popular tools

for Portugal nowadays still have this proximity with these programming languages, but they have lost space in introductory programming courses, becoming obsolete. Calango language was created to solve this problem for Portuguese speakers. It is a Portuguese-structured language with a dedicated IDE. Besides executing its own language, Calango IDE has a debugger for novices to keep track of the program variables. Its keywords and structure were inspired by the C language, which is a widely used language in introductory programming courses.

1.2 Justification

Professional programming languages have various supporting tools at their disposal, such as debuggers, dedicated IDEs, and interactive tutorials. Another relevant tool is the online judge, which helps both students and professors directly. [Silva et al. \(2020\)](#) made an exploratory research that gathered students' impressions about using Calango in an introductory programming course. Students expressed positive opinions about Calango and some students suggested the use of an online judge for it, such as the one they used for C language later in the course.

By automatically testing students' code, the online judge enables the professor to apply more activities with proper testing, and the students to have instant feedback on their code. In [Canedo, Santos and Leite \(2018\)](#), 57% of interviewed students from the University of Brasília (UnB) stated that the lack of practical exercises is negatively impactful when they are learning computer programming. If a professor does not use an online judge as part of his teaching methodology, either he will not test his students' code, or he will be limited by the number of the students' programs he can test. Therefore, this work proposes the development of an online judge for Calango. This online judge is tested through cycles of Action Research due to its suitability for educational scenarios as a research method because of its practical, participative, and collaborative nature.

1.3 Research Question

Notwithstanding that the literature has supported the hypothesis that online judges have a positive impact on students' programming learning, this work intends to prove the same hypothesis but considering the singularity of a teaching-learning methodology that uses Calango. However, since a positive impact is expected, the focus of this work is to clarify in which way students are influenced. Therefore, the research question is as follows:

How students' behavior and satisfaction are affected by the use of an online judge for the Calango language?

1.4 Aims and Objectives

In light of the advantages of having automated assessment through online judges and enforced by the literature, this work intends to develop an online judge for Calango. The specific goals are:

- Implementation of an online judge for Calango with an architecture consistent with what is presented in the literature and meeting requirements of a reliable web platform of this nature.
- Inspection of the already developed Calango software to implement improvements according to the experiences of use in class and to incorporate the source code into the online judge.
- Conduction of an experiment that applies the online judge in an undergraduate course that introduces computer programming to students.
- Analysis of surveys' responses to confirm if this online judge has a positive impact on students' opinions.
- Analysis of data from the online judge to uncover relevant behavior patterns.

1.5 Manuscript Organization

- **Chapter 2 - Background and Theory:** presents the concepts related to the theme of this work.
- **Chapter 3 - Methodology:** details the research and development methodology.
- **Chapter 4 - Calango Online Judge (COJ):** describes the technical and user details of the online judge.
- **Chapter 5 - Results and Discussion:** describes the results of the surveys and the data collected by COJ.
- **Chapter 6 - Conclusion:** presents conclusions of this work and suggests some possibilities for future works.

2 Background and Theory

This chapter addresses the theoretical framework that supports this work by reviewing the literature and explaining relevant concepts about programming education.

2.1 Meaningful Learning in Introductory Programming

The Meaningful Learning Theory proposed by [Ausubel, Novak and Hanesian \(1978\)](#), briefly explained, is the process when previous knowledge is used as a bridge for acquiring new knowledge. Ausubel defended the Cognitive Learning Theory, which states that information, in the form of knowledge, is hierarchically organized and stored in a "cognitive structure". This structure is a network of concepts already established by the learner.

Meaningful Learning happens if three conditions are met ([WANG, 2020](#)). First, the learning materials must have logical meaning by interconnecting subjects and gradually introducing new concepts. The second is learners' commitment and interest to learn meaningfully. The third one is the existence of previous knowledge in the cognitive structure that can establish connections with the new knowledge.

In introductory programming, meaningful learning can be achieved by allowing students to program first using their native language. This way, they rely on the knowledge already established in their cognitive structure (their native language) to acquire new knowledge (build algorithms). This methodology would also be supported by the fact that it is possible to trace parallels between cognitive mechanisms of programming and natural languages ([FEDORENKO et al., 2019](#); [PORTNOFF, 2018](#)).

[Roussel et al. \(2017\)](#) conducted three experiments in higher education where 294 students received varied academic texts in three different conditions: native language, foreign language, and foreign language with a translation into the native language. Results showed that foreign language presentation without explicit foreign language instructional support decreased both language and content learning.

In [Guo \(2018\)](#), non-English speakers reported a lot of different barriers when learning programming, such as: reading instructional materials, technical communication, reading code, writing code, and learning a foreign language at the same time as programming. An interviewed learner stated: *"My low level of English fluency badly affected my learning of a programming language at the beginning."*

2.2 Calango

Calango is a multiplatform educational software developed in 2012 that allows the construction, debugging, and execution of algorithms. The motivations for developing Calango were (FELINTO; GIROTTO, 2012): to reduce programming complexity; to provide a straightforward view of code execution; to facilitate user experience; to encourage good practices; to facilitate the transition to C language. Its main advantages are (SILVA et al., 2020):

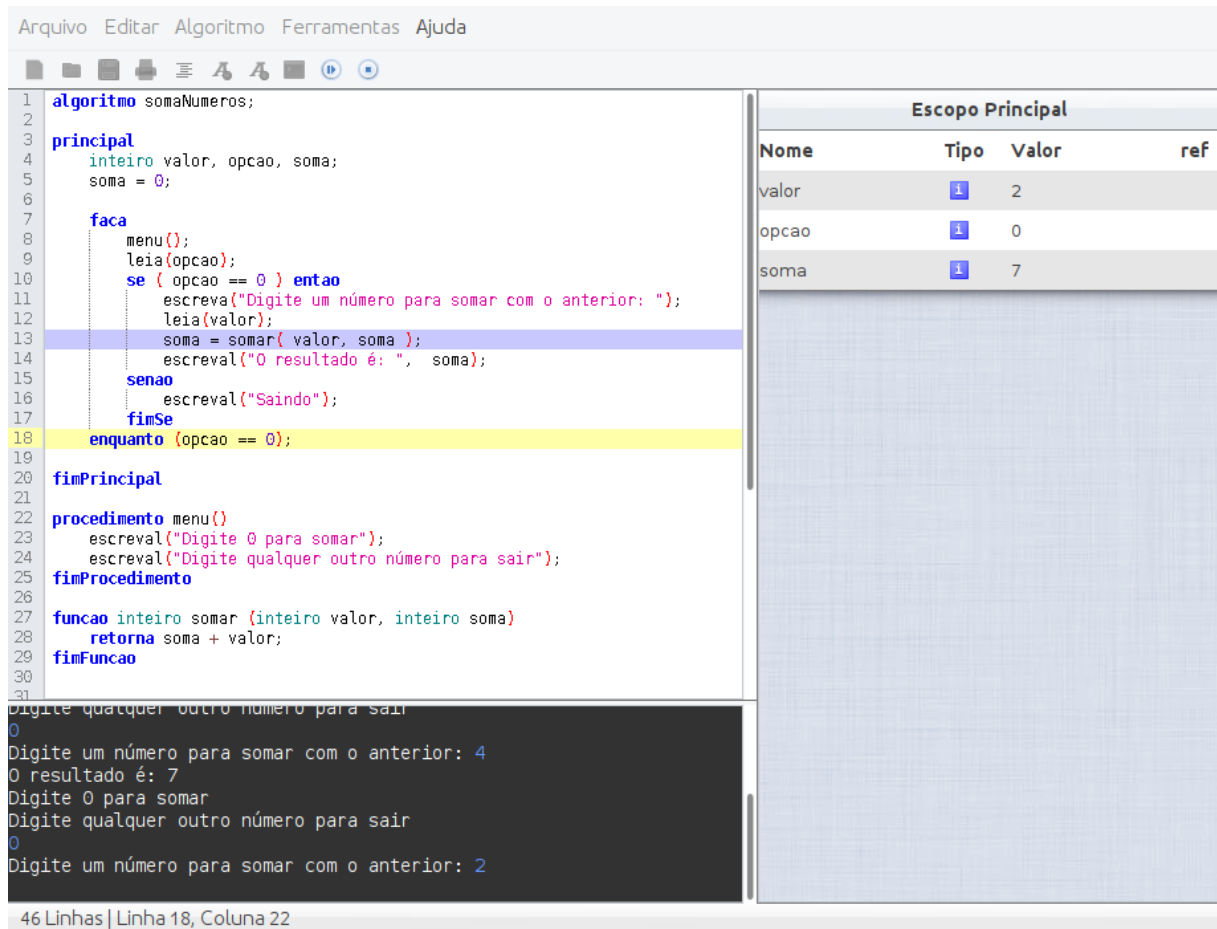
- It removes the foreign language barrier for Portuguese speakers.
- It is as simple as pseudocode. Nevertheless, it is executable.
- It has its own Integrated Development Environment (IDE) with debugging tools for novices.
- Errors are presented in a more instructive way and in Portuguese.
- It is a multiplatform tool that does not require installation.
- Because it is based on the C language, it facilitates transitioning from one to another.

Choosing C as a base language for Calango's development was not an arbitrary decision. In 2017, 657 students from the University of Brasília (UnB) were interviewed and 95% confirmed that they used C language in their introductory programming course (CANEDO; SANTOS; LEITE, 2018). In 2016, 218 colleges and 143 universities in 35 European countries were analyzed by a study and their key finding was that the introductory programming language that was most often taught in the 1st semester is C (45.7%) (ALEKSIĆ; IVANOVIĆ, 2016).

Calango was developed with Java and is composed of two main packages: the interface and the interpreter. The interface was developed with a graphical user interface toolkit called Java Swing. Figure 1 shows the appearance of the Calango IDE. On the top side, there is the toolbar with options related to file management, code editor, code execution, and assistance instructions. In the middle left side, there is the code editor followed by the console on the bottom. On the right side, there is the pile of scopes which is the debugging tool's visualization.

The toolbar is simple and presents a minimum number of options, as well as the whole interface. It is important to have in mind that Calango is not a substitute for C or other advanced languages, because it seeks to introduce basic concepts and focus on problem-solving. Therefore, as an introductory language, its syntax and interface do not need complexity, because other advanced features of an IDE can be introduced later in the course with C.

Figure 1 – Calango IDE



Source: the author

The interpreter was built upon the artifacts generated by the ANOther Tool for Language Recognition (ANTLR) parser generator. The language is first formally defined by the extended Backus–Naur form (EBNF). ANTLR takes the EBNF as input and generates GrammarLexer and GrammarParser as Java files (FELINTO; GIROTTO, 2012). The language and syntax definition is very close to the C language to reduce students' learning curve in introductory programming. Table 1 maps Calango's keywords to corresponding C language keywords. This similarity is also a differential point for Calango.

The interpreter is an entirely independent package that can be easily used by other Java programs. The interface was also built as an independent package although it depends on the interpreter to fully function. The interface connects to the interpreter through an API class that has entries for general execution or debugging mode.

2.2.1 Similar Tools

Since Calango was developed in 2012, there were many similar initiatives developed since then. Considering just the ones available for use nowadays, there are three direct competitors: [Portugol IDE 2.2](#), [Visualg 3.0](#) and [Portugol Studio](#). They are all IDEs

Table 1 – Corresponding Calango keywords for C keywords.

C keyword	Calango keyword
Main	
int main(void){...} int func() return	principal ... fimPrincipal funcao inteiro func() retorna
Logical operators	
&& !	e ou nao
Data types	
char* char double int 0, 1	texto caracter real inteiro logico
Read and write	
printf("...") printf("...\n") scanf ("...", var)	escreva("...") escreval("...") leia(var)
Conditionals	
if(){...} else switch(){...} case break	se() entao ... fimSe senao escolha() ... fimEscolha caso interrompa
Loops	
while(){...} do ... while() for(int i=0; i<=3; i++)	enquanto() faca ... fimEnquanto faca ... enquanto() para (i de 0 ate 3 passo 1)
Embedded Functions	
strcmp(const char* str1, const char* str2) strlen(const char* str) abs(int x) pow(double x, double y) sqrt(double x) toupper(char c) tolower(char c) M_PI	comparaTexto(texto str1, texto str2) tamanhoTexto(texto str) abs(inteiro x) exp(real x, real y) raizQuadrada(real x) maiusculoCaracter(caracter c) minusculeCaracter(caracter c) pi()

Source: adapted from [Silva et al. \(2020\)](#)

capable of executing their own variation of structured Portuguese-written pseudocode. These variations are very similar but they have their particularities.

[Portugol IDE 2.2](#) differs from Calango's syntax mainly in three points. First, the variable attribution is done by using the operator "<=". Second, commands are not followed by semicolons. Third, the command to print data to the console does not surround arguments with parentheses. One downside is that it does not support subprograms whereas Calango does. Although [Portugol IDE 2.2](#) is available for download, its last version is from 2006, suggesting that it is no longer maintained.

[Visualg 3.0](#) was developed before Calango and it is still maintained. The main difference is the variable attribution because it has inherited entirely from Pascal, differing only the variable types because they are written in Portuguese. Pascal does not appear in the list of the most recently used programming languages in introductory courses from Brazil, the USA, and countries from Europe ([CANEDO; SANTOS; LEITE, 2018](#); [ALEKSIĆ; IVANOVIĆ, 2016](#); [EZENWOYE, 2018](#)), therefore it has become obsolete for teaching computer programming to novices. Another downside is that Visualg is not supported in macOS, a widely used operating system.

[Portugol Studio](#) is the most recent tool of them all, it is still maintained, and it is closer to C language than Calango. However, when [Portugol Studio](#) was being developed, Calango was already being used in classes for years. Calango is a more mature educational tool because it has studies supporting its use in class with student approval ([SILVA et al., 2020](#)).

2.3 Online Judge

In the research conducted by [Malik \(2018\)](#), programming exercise questions and answers appeared as one of the most useful learning resources for 67% of the students. However, considering that testing and correcting programming exercises is a laborious process, professors rely on online judges to reduce the workload.

2.3.1 Definition

Online judges are a subcategory of a broader class of Automated Assessment Systems or Automated Grading Systems, which are not restricted to programming. They receive an input, usually a student activity, and return a result based on some evaluation criteria. Online judges receive source code as an input and they can run dynamic or static tests to check the correctness of the submitted source code.

The terminology "online judge" was first used in the literature by [Kurnia, Lim and Cheang \(2001\)](#) and it was defined as "an automatic programming assignment grading

system". Although it was the first time it was referred to as an online judge, similar systems had been already in use at that time in a series of different contexts. However, this marked the separation of online judges from other grading systems of general-purpose, and it gained its own segment.

According to [Wasik et al. \(2018\)](#), based on an extensive literature review, an online judge is a service that performs the following activities in a cloud: collects, compiles sources if needed, and verifies executability of resultant binaries; assesses solutions based on a set of test instances in a reliable, homogeneous, evaluation environment; computes the aggregated status and evaluation score based on the statuses and scores for particular instances.

Notice that these activities are not exclusive to online judges, and can be addressed to any grading system for programming exercises, or even other kinds of input. However, the key definition for online judges is that it is run in the cloud. This is what sets it apart from other grading systems categories alongside its exclusive use for programming assignments.

2.3.2 History

[Douce, Livingstone and Orwell \(2005\)](#) identified three broad generations of automated assessment systems for programming assignments: the first generation systems were forerunners but limited to particular computing laboratories, for professors; the second generation systems were command-line-based tools, sometimes supported by GUI interfaces; and the third, and current, generation systems are marked by being web-based and having more facilities, that is, online judges.

Automatic grading systems for programming assignments started back in 1961 at Stanford University, with an ALGOL grader for students' programs ([FORSYTHE; WIRTH, 1965](#)). Unsurprisingly, it was an offline solution considering that there was no internet at that time. By definition, it was not an online judge per se. Nonetheless, the first generation of automated assessment systems paved the way for online judges, since the idea is the same, simply differing on the implementation and features.

Online Judges emerged stronger on competitive programming contests. The evaluation on the cloud was necessary because many people would work on parallel in contests and it was necessary to share the same evaluation environment. One of the first popular online judges for competitive programming was the UVa online judge that was open to the public in 1997 ([REVILLA; MANZOOR; LIU, 2008](#)). UVa's website shows that in 1997 there were 4031 submissions, while in 2020 there were 1445197 submissions ([UVa, 2021](#)).

Other popular and more recent online judges include [Codeforces \(2021\)](#), [SPOJ \(2021\)](#) and [CodeChef \(2021\)](#). However, because these platforms focus on competitions,

they lack tools for an educational environment and managing a course class. Therefore, [Tonin and Bez \(2012\)](#) created the URI Online Judge with an interface designed for learners and educators. As an online judge, its functioning does not differ from the others previously cited. However, URI's interface and organization are more suitable for use in class.

More importantly, URI Online Judge has an advantage over other tools for use in class because it has an Academic module for professors to manage disciplines, homework, deadlines, acceptance criteria, and plagiarism detection ([BEZ; TONIN; RODEGHERI, 2014](#)). These features provide full monitoring of students' performance. By having this kind of visualization and control, educators can detect problems in the learning process and adjust methodology.

2.3.3 Advantages of Automated Assessment

The literature presents a variety of benefits coupled with the use of automated assessment in programming assignments. An online judge can increase students' interest in programming, programming skills, and overall performance ([WU et al., 2016](#)), besides leaving time for professors to focus on other educational activities. This kind of assessment saves the professor's time wasted with testing and allows students to "work anytime, anywhere and with immediate feedback" ([SKALKKA; DRLÍK; OBONYA, 2019](#)).

[Wang et al. \(2016\)](#) developed an online judge for a course that focuses on programming thoughts and methods. The teaching method was adapted to be contest-driven and practice-oriented. Students that participated in the adapted teaching method with the online judge exhibited higher enthusiasm and had their practical and comprehensive abilities more promoted than the control group.

In [Verdú et al. \(2012\)](#), the EduJudge system increased students' scores and satisfaction in comparison to students who did not use it and the interesting finding is that "the level of students' satisfaction does not depend on their computer skills levels". [Aleman \(2011\)](#) found a positive correlation (0.85) between the number of accepted submissions in the Mooshak online judge and students' final scores in the course. This correlation encourages studies about students dropping out from classes using online judge data, because it can perhaps predict final scores.

With higher interest and performance from students, the use of online judges also impacts positively the withdrawal rates. In [Wilcox \(2015\)](#), the average withdrawal rate dropped from 10.4% to 4.9% and students started submitting earlier in comparison with the traditional method. However, this study also raises a point of concern, because students work only until the judge tests are passed, missing subjective requirements.

The replacement of humans by machines is just a consequence of technology be-

coming more accessible. The use of an online judge is no different and it carries the same benefits other processes achieve when using automation. One of these benefits is the reduction of human failure. In the educational process, this has greater importance, because the assessment process is exhaustive and professors "may mark differently as they become fatigued as well as being affected by the order of marking" (HALEY et al., 2007). On the contrary, the online judge always gives the same output for the same input infinite times.

2.3.4 Important Features

Online judges can be used in a variety of contexts, such as competitive programming, education, online compilers, recruitment, and data-mining services (WASIK et al., 2018). In each context, there is the need for specific features, even though the problem description and test cases are basic features for online judges. In addition, they all share a security requirement that requires running submitted code in an isolated environment.

One popular isolation technique is sandboxing. This method is "a security mechanism for separating running programs, often used to execute untrusted source code" (YI; FENG; GONG, 2014). Sandboxing is not a very recent concept because it basically relies on virtualization. However, its implementation methods have evolved considerably. Nowadays, Docker is a widespread option for sandboxing online judges, because it is flexible, simplified, and ensures isolation (ŠPAČEK; SOHLICH; DULÍK, 2015).

Regarding what aspects the online judge can evaluate, there are a lot of possibilities, such as functionality, efficiency, testing skills, coding style, programming errors, software metrics, and design (ALA-MUTKA, 2005). Popular and free online judges usually focus on the dynamic and functional assessment draw on test cases. Static assessment is more difficult to automate as it can get subjective, but, even though it is not popular, it is possible to achieve.

In the educational context, there are other required features for supporting professors and students. Pieterse (2013) and Ullah et al. (2018) made an extensive review of online judges for automated assessment. Below are the features recommended by both of these works:

- **Well designed test cases:** the quality of the assessment is dependent on the quality of the test cases. They must cover every program limit, such as stop conditions and input exceptions. Both success and failure conditions must be tested, to ensure students covered all situations. One way to guarantee this coverage is to define test cases in parallel with the assignment requirements.
- **Detailed feedback:** submissions must transcend the correct and incorrect definitions and provide more information about the result so that students can locate

the error more easily. Varied grading results also allow professors to grade partially correct programs accordingly. If possible, the result should follow common grading metrics used by online judges.

- **Resubmission capacity:** even the experienced programmers are prone to errors, hence novices are not expected to get a fully corrected submission at their first attempt. Furthermore, resubmission encourages students to learn from errors by tracing their code, which is a day-to-day task in programming professions.

With regard to resubmission and detailed feedback, [Ihantola et al. \(2010\)](#) raises the importance of limiting maximum submissions to force students to think by themselves after their code received a failure result. Resubmissions can be managed by limited attempts or time penalty at every new attempt. Feedbacks should give only an idea of what went wrong but not give a full error message, because students may bypass local testing. The judge must be the code's final destination and testing and debugging should be done locally.

2.4 Chapter Summary

This chapter starts by explaining the Meaning Learning Theory and how it applies to programming education. Then, it describes the Calango platform and how it stands out among other Portugol tools. Lastly, it presents a wide literature review on online judges and establishes common ground for the features of this type of software.

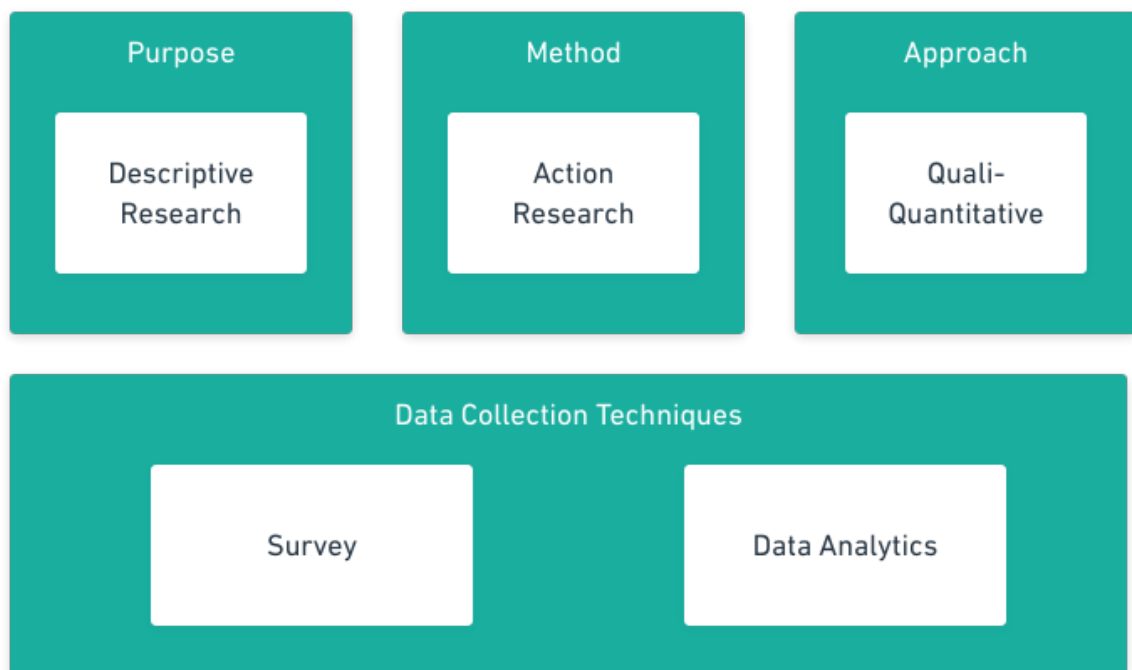
3 Methodology

This research is dependent on software development to collect results. Therefore, Section 3.1 explains how the research itself was conducted and its results evaluated, while Section 3.2 explains the methodology used for developing the online judge as a software solution.

3.1 Research Methodology

All aspects of the research methodology are shown in Figure 2. The descriptive purpose is a result of the exploratory work that has been done in [Silva et al. \(2020\)](#), where general aspects of Calango were addressed and from there arose the need for an online judge. The quali-quantitative approach is ideal because it is possible to run statistical analyses and subjective interpretations from empirical observations through action research, which provides great interaction between researcher and the object of study.

Figure 2 – Research Methodology



Source: the author

3.1.1 Action Research

The research method used was Action Research, which is "intended to solve practical problems of an individual or a group or an institution through planned intervention in the day-to-day working" (KHAN et al., 2018, p. 88). This method requires that both the researcher and a problem owner collaborate to solve a problem (EASTERBROOK et al., 2008). Action research has proven to be an appropriate and efficient methodology for conducting improvements in introductory programming courses (MALIK, 2018).

In this research, the professor and researcher worked together actively while conducting a course that used the online judge. Students were consulted and observed closely throughout the experiment. An experiment with the online judge is conducted within a class of the Faculty UnB Gama, one of the campuses of the University of Brasília. The course chosen introduces programming for freshman Engineering students. However, the course is not limited to students of its campus, making it possible for students from other courses to enroll.

Action Research is a cyclic process that seeks improvements in each iteration based on the knowledge acquired in the previous iterations. According to (COLLATTO et al., 2018), Action Research comprises four main phases: preliminary, to understand the problem; conduction cycle, to execute the course of action; meta-phase, to monitor the cycles; and the final phase, to communicate and publicize the research.

3.1.2 Surveys

A set of surveys were used at different times to collect data and analyze students' profiles and impressions. Google Forms was chosen to share the surveys with students. The surveys were applied at different times of the course because questions were dependent on what moment of their learning students were at.

For example, questions about expectations must be asked before they had any experience, while usability questions should be asked right after the use of the software. Lastly, questions about the methodology must be asked at the end of the course, because students can only confirm if it was a helpful experience after transitioning to the C language. Research questions are detailed in Appendix D.

The evaluation of software usability relied on the System Usability Scale (SUS), which was created in 1996 as a "quick and dirty" usability test (BROOKE, 1996). It is composed of 10 questions that approach different usability aspects and the final result is a measure of user perception about the software (DREW; FALCONE; BACCUS, 2018). This questionnaire is shown in Annex A.1.

SUS is a reliable scientific instrument that has been studied in many works, therefore it is a widely used measure of perceived usability (LEWIS, 2018). Because of that,

there were many works upon translations of SUS. The European Portuguese version of SUS was validated as a usability test instrument ([MARTINS et al., 2015](#)) and is presented in Annex A.2. Although in this work our participants are Brazilian Portuguese speakers, SUS has proven to be not sensible to minor wording changes ([BANGOR; KORTUM; MILLER, 2008](#)).

3.1.3 Data Analytics

The data stemming from the online judge was used to analyze student behavior through their submissions. For that, it was necessary data analytics techniques for extracting and processing data. The surveys evaluate the online judge through the students' point of view while the data analytics evaluates through students' results.

3.1.4 Research Tools

- Digital Libraries: [Google Scholar](#), [ACM Library](#), [IEEE Xplore](#), [Springer](#) and [Taylor & Francis Online](#)
- [Google Forms](#): web application to manage and share surveys.

3.2 Software Development Methodology

The software development methodology chosen was the Rapid Application Development (RAD). The next section defines what is RAD and explains how each of RAD's phases were used in this work.

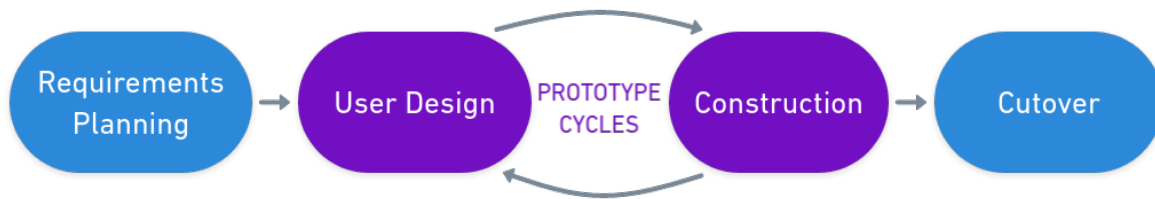
3.2.1 Rapid Application Development (RAD)

[Martin \(1991\)](#) first defined the Rapid Application Development as "a development lifecycle designed to give much faster development and higher-quality results than those achieved with the traditional lifecycle". Although RAD's definition places it very close to the Agile methods, RAD is more flexible and focused on software quality. Some of the main characteristics of RAD are ([BEYNON-DAVIES et al., 1999](#)):

- Low costs;
- High quality;
- Incremental prototyping;
- Rapidity of development;
- Highly interactive, low complexity projects;

RAD methodology phases are shown in Figure 3. The central point of RAD is the iterative cycle where prototypes are designed, implemented, and tested until a satisfying version is finalized. The iterative cycle can be achieved with non-functional responsive interfaces or with a functional software sample. We chose the latter to speed up the concluding phases.

Figure 3 – RAD Methodology



Source: the author

RAD methodology fits well this project because it adheres to Action Research and to the software development conditions. As explained in Section 3.1.1, Action Research is an iterative process as well as RAD. They both work with iterative cycles of improvement until the result is satisfying. Additionally, RAD is in accord with the short period of development and the fact that the software does not need to be impeccable and complete for the pilot test.

With incremental prototyping, we can have a Minimum Viable Product (MVP) at the pilot test and benefit from end-users feedback, iterating until we have a definitive version. Another reason for choosing the RAD methodology is the low complexity of the online judge as a software solution.

One of the requirements for using RAD is having small teams and developers with good knowledge of the technologies used. The only people involved in the development were the authors of this manuscript, and their skills determined the development tools and programming languages to be used.

3.2.1.1 Requirements Planning

Following the RAD methodology, preliminary requirements were elicited and later adjusted according to user feedback and testing. The elicitation techniques used were *Unstructured Interview*, *Introspection*, *Domain Analysis*, and *Prototyping*. Below are the definitions of each technique according to Zowghi and Coulin (2005) and how they were used in this work:

- **Unstructured Interview:** is a traditional, common, and conversational technique where the interviewer has limited control over the discussions because he does not follow a predetermined agenda or list of questions. The chosen interviewee was a

professor with experience in teaching Calango. Questions were made via email and messaging apps, which helped to keep the requirements' traceability.

- **Introspection:** is a technique where requirements are elicited based on what the developer thinks the end-users want and need for the system. In this case, the introspection figures were students and teachers that would use the online judge.
- **Domain Analysis:** consists of analyzing similar applications and related documentation to come up with requirements for the system being developed. The URI Online Judge is a similar application that served as a reference since it is free, open for testing, and already used by the interviewed professor.
- **Prototyping:** is a simple experimental system model that is typically developed using preliminary requirements or existing examples of similar systems. Requirements were added and adjusted by relying on the RAD's prototype cycle.

These techniques were chosen in accordance with the RAD methodology and considering: the short development time, the limited number of stakeholders, similar systems available for testing, and the developer's familiarization with the domain.

3.2.1.2 Prototype Cycles

The first cycle started with an MVP of the online judge that was tested in an introductory programming course of the University of Brasília, as a pilot test. While students were submitting code and being evaluated by the professor, the judge was being improved. Users' feedback and observations played an important part in identifying problems and points of improvement.

The cycles did not have predefined periods and tasks, as they took advantage of RAD's flexibility. They kept going until the professor finished activities with Calango and proceeded to C language. Some new requirements or improvements identified were treated right away, while others were treated in the next cycle or the Cutover phase that generated the final software version.

3.2.1.3 Cutover Phase

The final phase was retained for final improvements, documentation, and disclosure. Less important details are taken care of, such as aesthetics and code improvement. Since the software was constantly changing, there was no definitive documentation, and specific details are covered in this phase, including user guides. Finally, the software is released for a broader audience.

3.2.2 Development Tools

- [Git](#): free and open-source distributed version control system.
- [GitHub](#) & *GitHub Packages* & *GitHub Actions*: free web interfaces for hosting the source code, hosting Docker images, and auto-deploying code changes, respectively.
- [Docker](#) & *Docker Compose (Version 1.26.0)*: free and open-source platform services for virtualization and container orchestration, respectively.
- [Django Framework \(Version 3.1.4\)](#): free and open-source Python framework for rapid web development.
- [Spring Boot \(Version 2.4.0\)](#): free and open-source Java framework for web development based on design patterns.
- [Maven](#): free and open-source software project management and comprehension tool.
- [NGINX \(Version 1.19.0\)](#): free and open-source lightweight reverse proxy server and load balancer.
- [PostgreSQL \(Version 13.0\)](#): free and open-source Relational Database Management System (RDBMS).
- [DigitalOcean](#): Infrastructure as a Service (IaaS) platform for hosting software.

3.3 Activities Timeline

This work is developed in a period of 13 months. Figure 4 depicts the activities of each month from October 2020 until November 2021. Action Research's cycles are marked by the application of 3 surveys each, and RAD's prototype cycles happen between November 2020 and September 2021.

3.4 Chapter Summary

This section depicts both methodologies used in this work. Action Research is used to follow the experiment closely and RAD is used to develop the online judge. They are both cyclic and well-defined processes that fit this research scenario and objectives. Furthermore, surveys and data analytics techniques are used as research instruments to gather information during the experiment.

Figure 4 – Activities timeline



Source: the author

4 Calango Online Judge

This chapter presents the technical details of the Calango Online Judge (COJ) as a software solution. Section 4.1 lists the final requirements after the RAD's iterations, Section 4.2 depicts the architecture's modules and Section 4.4 describes the stages and infrastructure for delivering COJ to users. The source code developed in this work is available as open-source code and the repositories are linked in Appendix A.

4.1 Requirements

The first requirements were elicited in the Requirements Planning phase. Then, the existing ones were improved and new ones were added according to user feedback at the Prototype Cycles. Both functional and non-function requirements were listed and are shown in the following sections.

4.1.1 Functional Requirements

The functional requirements modeling made use of User Stories, which are "a popular method for representing requirements, often using a simple template such as "As a <role>, I want <goal>, [so that <benefit>]" (LUCASSEN et al., 2016). Table 2 shows COJ's functional requirements as user stories grouped by epics. The "Pre-traceability" column points to the elicitation techniques that generated the requirement, whereas the "Post-traceability" column points to the finished interface that satisfies the requirement. Elicitation techniques are ordered in a way that the first technique was responsible for discovering the requirement and the following were used to refined acceptance criteria.

4.1.2 Non-functional Requirements

For classifying non-functional requirements, it was used the quality model from ISO/IEC 25010 (2011), which categorizes product quality properties into eight characteristics (functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, portability) that are composed of a set of related sub characteristics. Some requirements may attend more than one quality sub characteristic, but only the most satisfied sub characteristic is listed. Non-functional requirements worked as a complement to acceptance criteria for functional requirements.

Table 2 – COJ’s functional requirements

Epic	Role	User Story	Pre-traceability	Post-traceability
Authentication	User	I want to log in with my username and password so that the system can authenticate me and I can trust it.	(DA)	Section 4.3.1
	User	I want to be able to request a new password so that I don’t permanently lose access to my data if I forget it.	(IW)	Section 4.3.1
Lists	Student	I want to access my lists so that I see evaluative problems and the list’s deadline.	(DA)(IN)	Section 4.3.3
	Student	I want to see my list’s conclusion percentage so that I know what grade I will receive.	(IN)	Section 4.3.3
	Professor	I want to create lists of problems so that I can join similar problems in one evaluative assignment.	(DA)	Section 4.3.3
	Professor	I want to edit lists of problems so that I can correct their name or start/due dates in case they are wrong or outdated.	(IN)	Section 4.3.3
	Professor	I want to schedule lists so that students can submit code to the list only during the time between the start and due date.	(DA)(IW)	Section 4.3.3
	Professor	I want to see my lists and their schedules so that I can review their information and keep track of their status.	(DA)(IN)	Section 4.3.3
	Professor	I want to have different schedules for the same list so that I can assign different deadlines to each class.	(IN)	Section 4.3.3
Problems	Student	I want to access problems and test cases so that I can elaborate my algorithm correctly.	(DA)	Section 4.3.5
	Professor	I want to create problems and test cases so that my students can submit code and have them automatically reviewed.	(DA)(IN)	Section 4.3.5
	Professor	I want to have problems outside lists so that my students can practice beyond evaluative problems.	(P)(IW)	Section 4.3.5
	Professor	I want to see problems grouped by subject so that I can find problems faster to add to my new lists.	(IW)	Section 4.3.5
	Professor	I want to test my problems with code and have detailed feedback so that I ensure that my test cases are correct.	(P)	Section 4.3.5
	Professor	I want to edit my problems and their test cases so that I correct them in case there is any problem.	(P)	Section 4.3.5
Results	Student	I want to see general statistics of my submissions so that I can follow my progress.	(IN)	Section 4.3.2
	Professor	I want to see general statistics of my students’ submissions so that I can monitor students’ progress.	(IN)(IW)	Section 4.3.2
	Professor	I want to see the lists’ results question by question so that I can see how students are doing in each question.	(DA)(P)(IW)	Section 4.3.4
	Professor	I want to download lists’ results so that I assign students’ grades more easily.	(IN)(P)	Section 4.3.4
Submissions	Student	I want to submit code so that I can solve problems.	(DA)	Section 4.3.5
	Student	I want to receive a submission status so that I know if I solved the problem.	(DA)	Section 4.3.6
	Student	I want to know what each submission status means so that I understand why my code is failing.	(P)(IN)(DA)	Section 4.3.8
	Student	I want to see the submission’s code so that I identify what triggered the submission’s status.	(DA)	Section 4.3.6
	Professor	I want to see a student’s submission so that I can see his code to help him when he has difficulties with some problem.	(P)(IN)	Section 4.3.6
Classes	Professor	I want to group students by class so that I can assign different activities and deadlines to each class.	(IN)	Section 4.3.7
	Professor	I want to remove students from classes so that they no longer access my class data.	(IN)	Section 4.3.7
	Professor	I want to add students to my class by email so that they receive an invitation email.	(IW)	Section 4.3.7
	Professor	I want to deactivate a class so that students lose access to this class and they can be enrolled in another one in the future.	(IN)	Section 4.3.7
	Professor	I want to see my students in their class so that I can review their information and keep track of their status.	(IN)(P)(IW)	Section 4.3.7
	Professor	I want to see my inactive classes separately so that my active classes stand out to manage them more easily.	(P)(IW)	Section 4.3.7

D=Domain; IN=Introspection; IW=Interview; P=Prototyping

Source: the author

Table 3 – COJ’s non-functional requirements

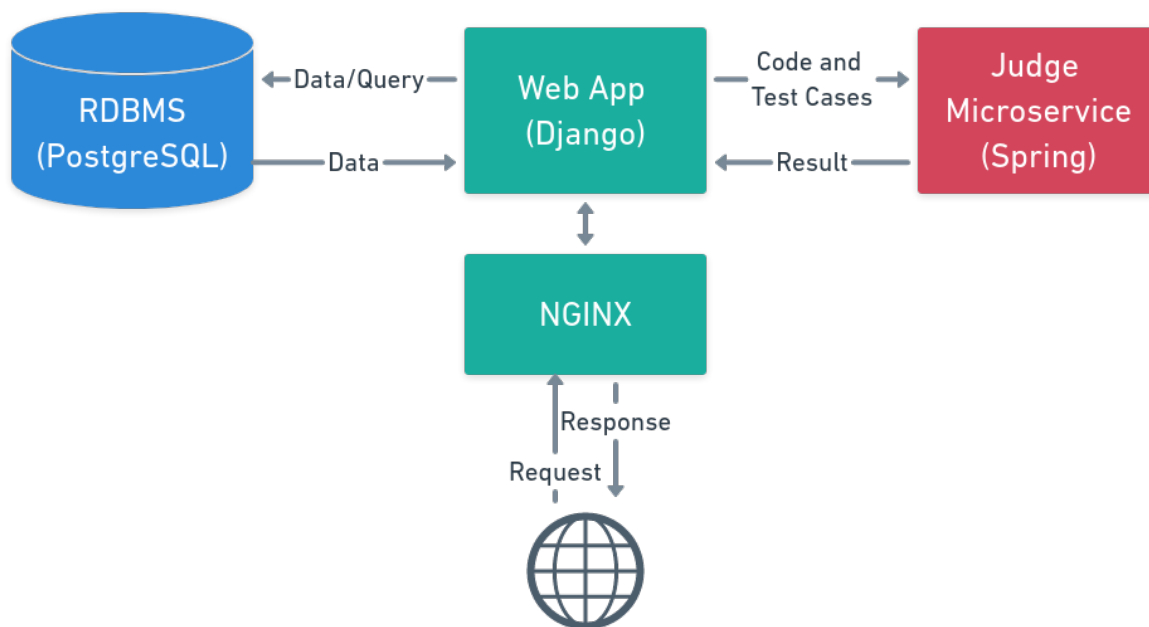
Requirement	Characteristic	Sub-characteristic
Isolate the judging module as a service in container sandbox	Maintainability	Modularity
Expire sessions after a browser closes to hinder access of unauthorized users.	Security	Confidentiality
Execute three attempts of running submitted code in case of an unexpected failure.	Reliability	Recoverability
Have daily backups and straightforward implementation through containers to quickly recover from failures.	Reliability	Recoverability
Parallelize the judging process in order to have simultaneous submissions.	Performance Efficiency	Time Behaviour
Do not block the user interface while code is under review and explicitly present status to the user.	Usability	Learnability
Limit submitted code to 5 seconds of execution and kill the running program when the limit is exceeded.	Recoverability	Fault Tolerance
Design a user experience similar to online judges for C language to have a smooth transition from one to another.	Usability	User Interface Aesthetics
The user interfaces must avoid the use of loanwords to be coherent with the methodology proposal, except URLs and the platform’s name.	Usability	Learnability
Each problem must have a detailed description of inputs and outputs containing: the type of data, what it represents in the context, and text formatting.	Usability	Learnability
Programming code must be avoided when it is possible to achieve the same functionality with SQL queries.	Performance Efficiency	Time Behaviour
Pages should require less than six clicks to be reached and the user must be visually informed of where is he on the platform.	Usability	Operability
The web app must work on Firefox and Chrome for Desktop besides providing basic functionality on mobile devices.	Portability	Adaptability

Source: the author

4.2 Software Architecture

COJ's architecture is composed of two main modules: the web application and the judge microservice. The web application is the only module in the system where end-users have direct interaction with the whole system and has a dedicated database. The judge service is a microservice and it is responsible to judge Calango source code and test cases, returning results back to the web application. Figure 5 shows an overview of the system's modules and how they interact. Appendix B shows some software artifacts that are referred to in the next sections.

Figure 5 – COJ Architecture



Source: the author

Creating the judge as a decoupled service allows the use of suitable programming languages for each service and parallelizes the judging process. The service-oriented design provides modularization, promotes loose coupling, allows applications to be reused, and permits an incremental approach (SERRANO; HERNANTES; GALLARDO, 2014).

NGINX acts as a reverse proxy and serves static files. Since the web application runs in port 8000, NGINX forwards requests from port 80 to port 8000. Also, Django does not serve static files in production, therefore NGINX has this responsibility. Lastly, Django provides out-of-box integrations with a set of Relational Database Management Systems (RDBMS), and PostgreSQL was chosen because of its suitability to system requirements and the experience of the programmer in working with it.

4.2.1 Judge Microservice

The microservice was developed with Spring and has an API for interacting with other services. It receives as inputs Calango source code and the test cases that will be tested one by one for the same source code. For that, the microservice needs to consume Calango's interpreter package.

Calango Interpreter is part of the original Calango source code, but it was built as a separate package from Calango IDE. The same logic was applied to COJ when using the interpreter, by consuming the interpreter package in the Spring application. Before the development of the online judge modules, it was necessary to study this source code. This also gave the opportunity to improve or correct some bugs in this source code.

It was necessary to do reverse engineering to recover Calango's source code from the distributed JAR file because its original remote repository went offline. After that, the code was reorganized in order to become executable and [Maven](#) was used as a build automation tool. Maven facilitated the project modularization because it is possible to join different repositories in one project by using the dependency declaration of external components. This way, there is one interpreter repository used by both Calango and the microservice.

The microservice receives submissions from the Web App through an HTTP POST request with a JSON body, transforms the JSON into Java objects, sends these objects to the interpreter to be run as a Calango program, and waits for the program's outputs. The interaction between the Web App, the microservice, and the interpreter is shown in Figure 24 at Appendix B. Below, there is an example of a JSON input that the microservice could receive from the Web App.

```
{
  "code": "algoritmo converteParaMaiusculo;\n\nprincipal\n\
    ↪ tcaracter letra;\n\n\tleiaCaracter(letra);\n\n\tescreval
    ↪ (maiusculoCaracter(letra));\n\nfimPrincipal\n",
  "cases": [
    {
      "input": ["a"],
      "output": "A\n"
    },
    {
      "input": ["b"],
      "output": "B\n"
    },
    {
      "input": ["c"],
```

```

        "output": "C\n"
    }
]
}

```

The source code above receives a letter as an input and prints it as a capital letter. Considering the source code above is correct, given the test cases, the following JSON response would be sent by the microservice:

```

{
  "code": 1,
  "message": "ACCEPTED",
  "errorMessage": "No error message"
}

```

The command `"escreval"` prints some content with a line break at the end. Alternatively, the command `"escreva"` prints content without the line break. The test cases in this example expect a line break at the end of each output. If we changed the command `"escreval"` to `"escreva"` the microservice would return the following JSON:

```

{
  "code": 3,
  "message": "PRESENTATION_ERROR",
  "errorMessage": "Expected: A\n Actual: A"
}

```

The judge service has six possible results, and they are all returned in the above JSON format. The results are detailed in Table 4.

4.2.2 Web Application

The web application was developed with Django, which has a plethora of ready-to-run functions and libraries at the developer's disposal. For example, by just applying some settings, Django manages user authentication and authorization by itself. Apart from that, there are a lot of open-source third-party packages for Django that can be added to your own Django project. The ones used in COJ were:

- **django-crispy-forms**: allows managing and customizing forms with straight-forward Python and template tags.
- **django-ckeditor**: transforms simple text fields into rich text fields with markup. It was added to the question's creation form field.

Table 4 – Results returned by the Judge Microservice

Result	Description
ACCEPTED	The code's output and test cases' outputs were identical.
PRESENTATION_ERROR	The code's output is correct but wrongfully formatted due to letter case, spaces, or line breaks.
WRONG_ANSWER	The code's output has differed entirely or partially from test cases' outputs.
RUNTIME_ERROR	The judge was able to run the source code, however, it stopped unexpectedly or finished execution without presenting an answer.
COMPILATION_ERROR	The judge was not able to run the source code because of a syntactical error.
TIME_LIMIT_EXCEEDED	The judge was able to run the source code, however, it took more than five seconds to finish.

Source: the author

- **django-q**: allows multiprocessing procedures. It is used for sending mass emails to students and calling the judge microservice without blocking the web application.

The third-party package `django-q` plays a vital part in the judging process. It takes care of sending submissions asynchronously to the judge service through async tasks. Each task is saved in the PostgreSQL database alongside its results and error message, this way it is possible to investigate what happened with a submission that got an unexpected result.

Django-q stores successful, failed, and queued tasks. By default, `django-q` reruns failed tasks infinite times until it is successful, but the number of attempts can be changed by the developer. For example, if the judge service is unreachable, the task will fail and, after a configured amount of time, the task will be rerun.

The web application interacts directly with PostgreSQL. However, due to Django's Object-Relational Mapper(ORM), it is not required that the developer writes raw SQL queries. This makes the application more secure because it protects queries from SQL injection, which can insert malicious data into the database. Therefore, the database modeling is slightly different from a traditional SQL design, since it is done using Django's models but it follows the same rules for database modeling. The entity-relationship diagram for the web app is shown in Figure 22 at Appendix B.

However, it was necessary to understand how the database was disposed at PostgreSQL because information for the data analytics would be retrieved through SQL queries. Figure 23 at Appendix B shows the database schema automatically generated through Django's ORM. The SQL queries for retrieving data analytics information are

listed on Appendix C.

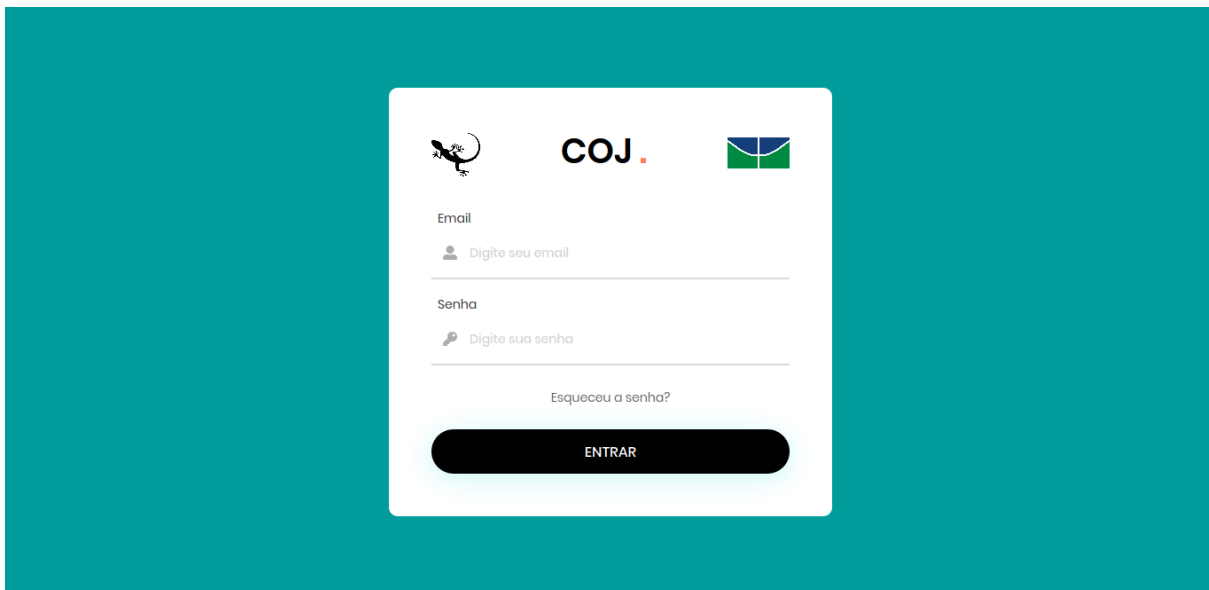
4.3 User Interfaces

While authentication and database queries are handled by Django, COJ has its own functionalities and interfaces, even though it benefits from the framework functions. This section details COJ's interfaces, how they work in the backend, and their business logic.

4.3.1 Authentication

The first page of COJ's is the login page shown in Figure 6. If an authenticated user types any URL, he will be redirected to this page. From there, the user can either log in or recover his password. It is not possible to self sign up. Professors are registered by the administrator and students are invited by their professors through email.

Figure 6 – Login page



Source: the author

4.3.2 Home

The first link in the sidebar is the home page and it is where professors and students are redirected after login. For both students and professors, there is some statistical information about the submissions and the list of problems as shown in Figure 7. Students see the data about submissions in their current class, while professors see data from his active classes. If he wishes to see separated statistics, he must go to the classes page.

Figure 7 – Home page (professor's view)



Source: the author

Students can only see data related to their current class. This limitation takes into consideration students that failed the course and is doing it for a second time. This way they cannot see past submissions and problems, which would be an advantage over other students. Besides the statistics from Figure 7, students see their average attempts per question and his class's average as well.

4.3.3 Scheduled Lists

The second link of the sidebar takes users to the scheduled lists page which is shown in Figure 8. Students have the same view except that instead of the class name in the last column of lists, they see if the list is closed or open according to the start date and due date.

In the backend, there is a difference between lists and their schedules. Lists are merely a group of problems. Schedules are used for assigning lists to a class with a start and due date. For example, a professor can create a list called "Repetition Structures" and schedule it for class "A" with a due date of five days. He can schedule the same list for class "B" with a due date of seven days.

Each class of the example would have its own results for the same list with different deadlines. This data modeling requires the professor to create a list just one time and have different deadlines for different course classes. Scheduled lists appear to students once the start date and time have passed, and submissions are no longer accepted after the due date. However, they still can see the problems description and test cases, besides their results.

Figure 8 – Scheduled lists page (professor's view)

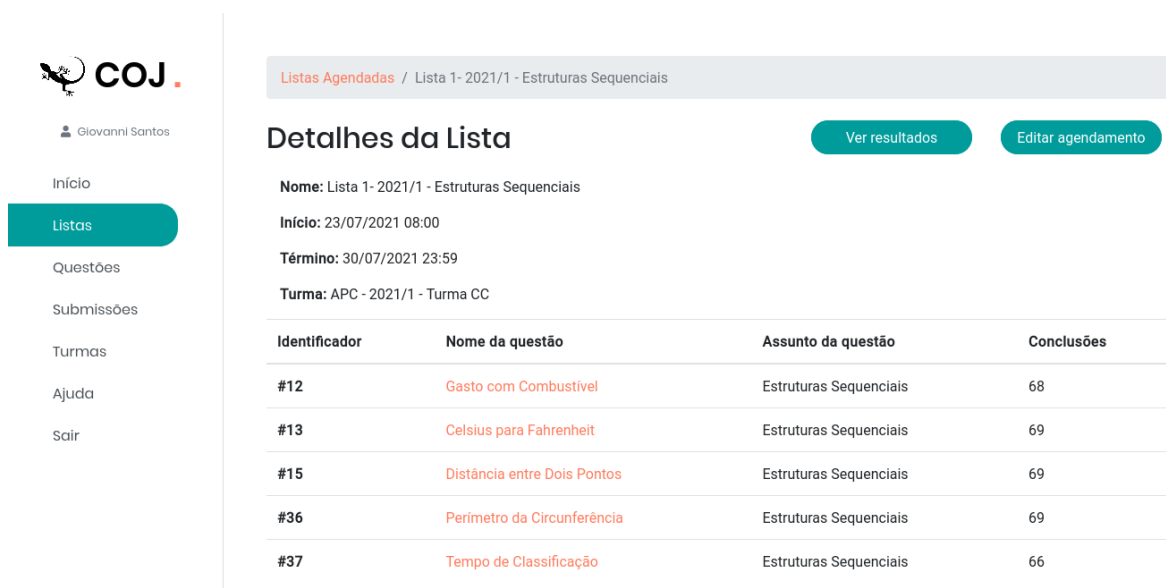


Nome	Início	Término	Turma
Lista 5 - 2021/1 - Vetores	20/08/2021 08:00	28/08/2021 23:59	TESTE - 2021/1 - Turma Mnt
Lista 5 - 2021/1 - Vetores	20/08/2021 08:00	28/08/2021 23:59	APC - 2021/1 - Turma CC
Lista 5 - 2021/1 - Vetores	20/08/2021 08:00	28/08/2021 23:59	APC - 2021/1 - Turma DD
Lista 5 - 2021/1 - Vetores	20/08/2021 08:00	28/08/2021 23:59	APC - 2021/1 - Turma DD-2
Lista 4 - 2021/1 - Modularização	13/08/2021 08:00	21/08/2021 23:59	TESTE - 2021/1 - Turma Mnt
Lista 4 - 2021/1 - Modularização	13/08/2021 08:00	21/08/2021 23:59	APC - 2021/1 - Turma CC
Lista 4 - 2021/1 - Modularização	13/08/2021 08:00	21/08/2021 23:59	APC - 2021/1 - Turma DD
Lista 4 - 2021/1 - Modularização	13/08/2021 08:00	23/08/2021 23:59	APC - 2021/1 - Turma DD-2
Lista 1 - 2021/1 - Estruturas Sequenciais	09/08/2021 14:30	23/08/2021 23:59	APC - 2021/1 - Turma DD-2
Lista 2 - 2021/1 - Estruturas Condicionais	09/08/2021 14:30	23/08/2021 23:59	APC - 2021/1 - Turma DD-2

Source: the author

When the user clicks on a list name, he is taken to the page's detail page shown in Figure 9, where the list's problems are listed. Moreover, students see their status on each problem and their overall score, while professors see the number of conclusions of each problem. Professors also see buttons for editing the schedule and to go to the result's page.

Figure 9 – Schedule's detail page (professor's view)



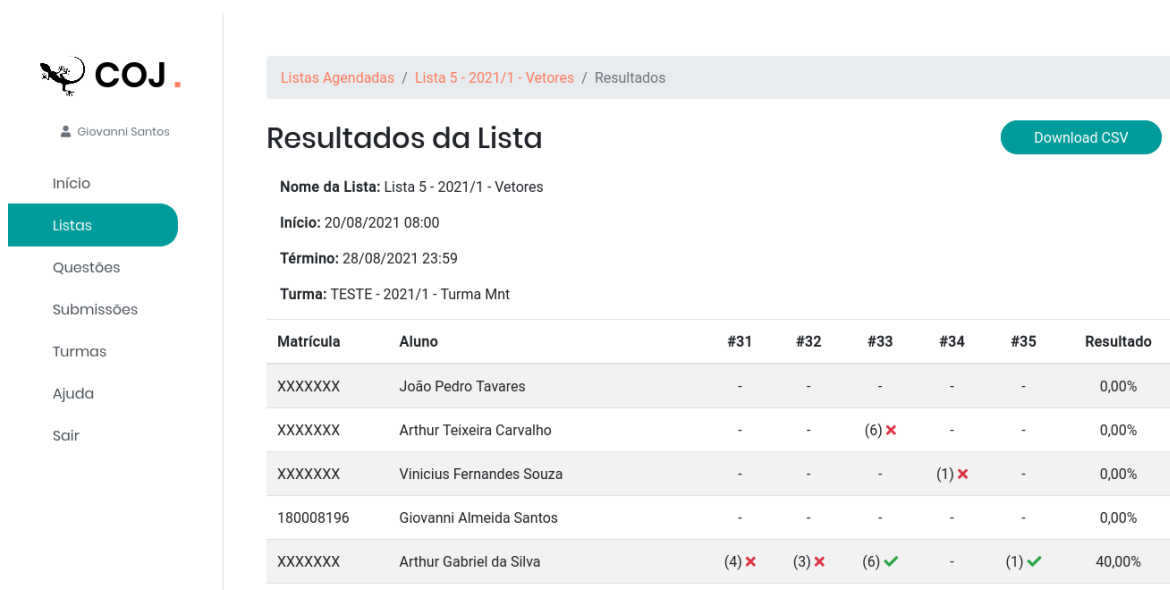
Identificador	Nome da questão	Assunto da questão	Conclusões
#12	Gasto com Combustível	Estruturas Sequenciais	68
#13	Celsius para Fahrenheit	Estruturas Sequenciais	69
#15	Distância entre Dois Pontos	Estruturas Sequenciais	69
#36	Perímetro da Circunferência	Estruturas Sequenciais	69
#37	Tempo de Classificação	Estruturas Sequenciais	66

Source: the author

4.3.4 List Results

Professors have a dedicated page for seeing students' results that is shown in Figure 10. This page shows all students from the class for which the list was scheduled, followed by their result on each problem. Icons indicate if there is an accepted submission followed by numbers which are the number of submissions for that problem. In the last column, there is a percentage of conclusions indicating how many problems the student concluded for this scheduled list. Also, professors have an option to download the results as a CSV file.

Figure 10 – Schedule's results page



Source: the author

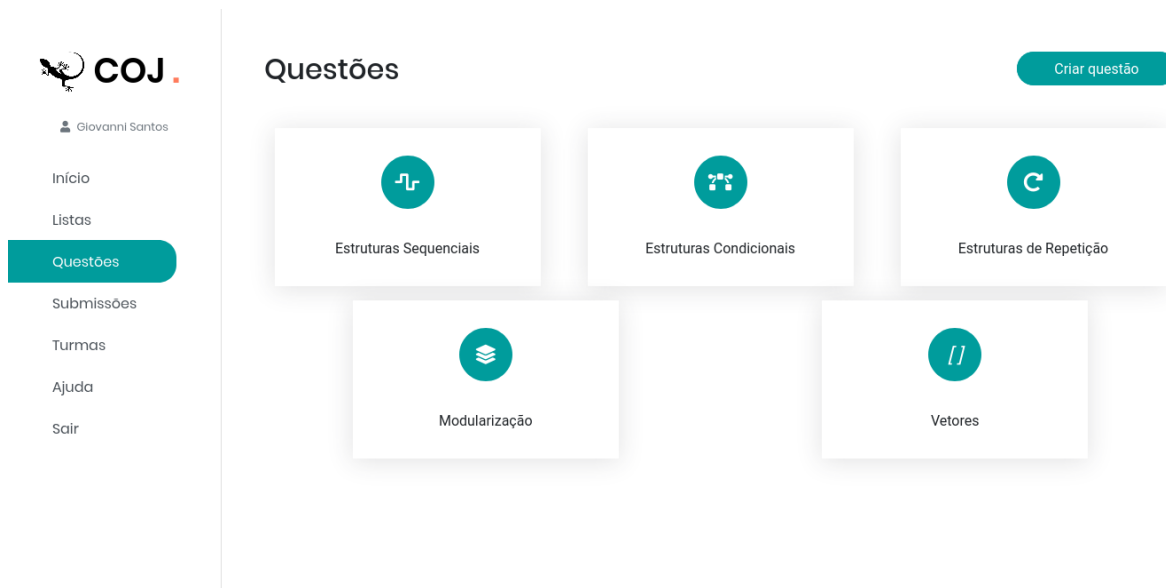
4.3.5 Problems

Questions are divided into 5 categories: Sequential Structures, Conditional Structures, Repetition Structures, Modularization, and Vectors as shown in Figure 11. Professors can view all problems registered in the system and create new ones, whereas students can only see non-evaluative problems from this page because evaluative problems are only available to them through lists.

The problem's details interface shown in Figure 12 is similar for students and the professor. Professors view a pencil icon to edit the question and a button to test the problem by submitting a solution. This functionality is important to check if test cases are correct.

Students have the submit button and a limited view of test cases since there are hidden ones to avoid deceit and to stimulate students to solve possible scenarios by themselves. If the problem's list has passed the due date, a "Closed" message replaces the

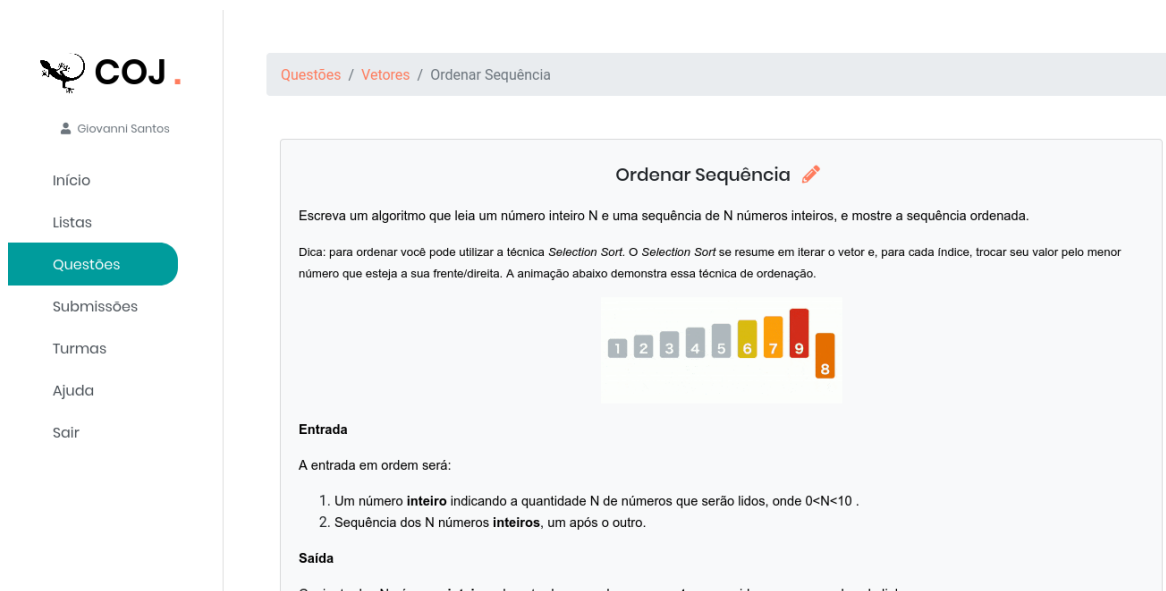
Figure 11 – Problems page



Source: the author

button. If there is already an accepted submission, the button is replaced by a "Concluded" message, preventing a new submission. The submission URL is also protected in case students try to access the submission page directly.

Figure 12 – Problem's details page



Source: the author

4.3.6 Submissions

Submissions' page interface is shown in Figure 13. Submissions have a time of submission and judging. It is important to differ both because in case of failures there

is a command to rejudge submissions. If this happens, the judging time will be updated accordingly.

Students see all of their own submissions referring to their current class, for either evaluative or non-evaluative problems. They are taken to this page right after submitting code to a problem. The result status shows "Waiting" until the microservice returns a result, which is shown by refreshing the page. Professors have a search box at their disposal to search for submissions that are from their active classes.

Figure 13 – Submissions page

Identificador	Nome	Questão	Submetido em	Julgado em	Resultado
#2572	Geovana Ramos Sousa Silva	Tempo de Classificação	21/07/2021 10:10	21/07/2021 10:10	Aceito

Source: the author

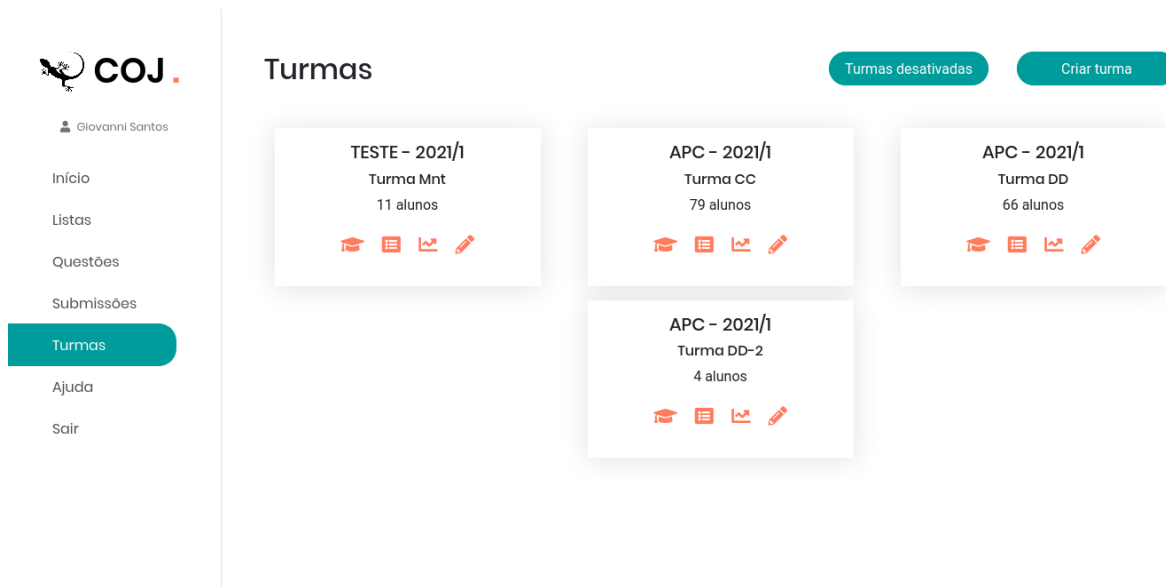
Submissions have an identifier, in case a student wants to refer to it when talking to the professor. It is also possible to check the submission code for both user types. For professors, it is an important feature to check how their students are coding and for manual analysis, if necessary. Submissions are linked to a question and a list. If a student submits to the same question that is linked to different lists, he will have a different result for each list.

4.3.7 Classes

The classes page is exclusive for professors and is shown in Figure 14. From there, it is possible to create a new class, edit class, deactivate class, add students, remove students, view the list of students, view the list of activities (lists and non-evaluative problems), see class's statistics, and go to inactive classes.

Once a professor creates a course class, he can add as many students as he wishes at the same time. The form receives students' full names and registration numbers in a CSV format with a ";" separator. This method was chosen based on the University of

Figure 14 – Classes page



Source: the author

Brasília (UnB) system, which presents classes in this format. This way, professors can copy the list of students from the University system and paste it into COJ.

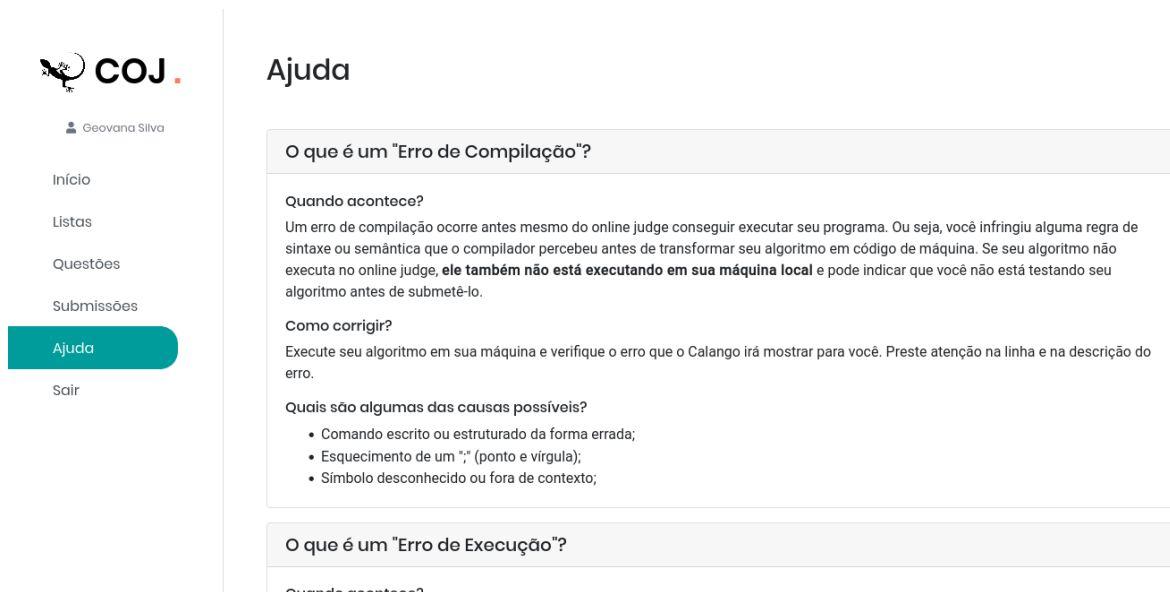
After submitting the list of students, they will be added to the class in a background process, because it is a long-running task. COJ sends an invitation email to students' institutional email and allocates them to a class selected by the professor. UnB's institutional emails are formed by the registration number, therefore professors only need to inform name and registration number. If the professor informs an email, emails are sent to the email informed instead of the institutional.

During deactivation, students are set to inactive users, and they lose access to COJ. However, their data is kept to preserve their course history. This happens because COJ displays data of an active class for students. If they are not in an active class, there is nothing for them to see in COJ. Once they are allocated to a new class, they regain access.

4.3.8 Help

This page is shown in Figure 15. It has only static content and it is visible and identical for both students and professors. For each result from the judge, there is a general explanation of when it happens, steps to correct it in case it is an error, and what are the possible causes.

Figure 15 – Help page



Source: the author

4.4 DevOps

Since this project is service-oriented, it has two remote repositories. One for the web platform and one for the judge service. Both repositories follow the same rules for continuous development and have the same configuration, except language-specific settings.

Continuous deployment was implemented through GitHub Actions and the infrastructure is hosted by DigitalOcean. Continuous deployment was important to advance quickly in the Prototype Cycle because new modifications came as fast as possible to testing users.

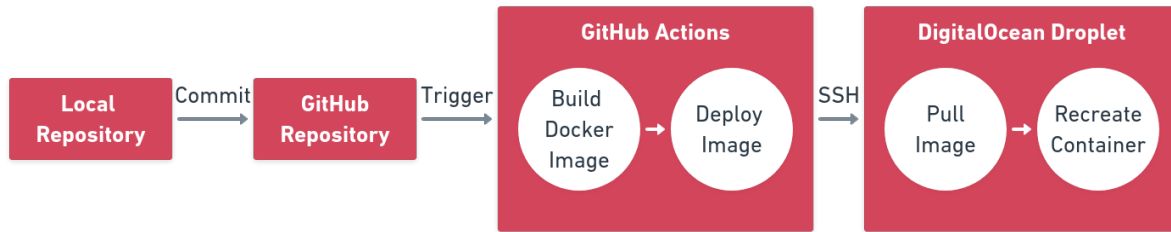
Docker and Docker Compose were chosen for supporting this development pipeline with a container-based approach. Service-oriented architectures benefit from Docker because a container image turns a given service into a black box, only exposing its API in exchange for resources, while Docker Compose can orchestrate any number of services with guaranteed low coupling because containers are sealed from one to another (GOUIGOUX; TAMZALIT, 2017).

4.4.1 Pipeline

The continuous deployment pipeline is shown in Figure 16 and starts with a commit from the local repository to the remote repository. Once the commit reaches the remote repository, GitHub automatically triggers GitHub Actions. This behavior is enabled by adding workflow files to the `.github` folder.

These files detail the pipeline and GitHub Actions jobs. In both repositories, the

Figure 16 – Deployment pipeline



Source: the author

first job is to build the Docker image using the existing Dockerfile in each repository. Then, the built image is sent to GitHub Packages, which is a hosting service for software packages.

Considering that images hosted in GitHub packages have open access, it is important to take care of some security details. Credentials and environment variables cannot be shipped with the image. Therefore, all of the sensitive information is passed to the container at the running time, this way the image does not carry confidential information.

Once the image is ready, the next job connects to the DigitalOcean server (called droplet) through Secure Shell (SSH). The server already contains the project's Docker Compose file pointing to the images hosted by GitHub Packages. Considering the last job has created a new image, it is necessary to update the server's images by calling *docker-compose pull*. Then, containers are recreated with *docker-compose up*.

4.4.2 Infrastructure

The server configuration was chosen according to the number of expected users in the experiment, which was around 60 users. DigitalOcean's servers are called droplets and developers can choose any amount of resources they desire. For the experiment, a basic droplet of 1vCPU, 1 GB RAM, and 25 GB SSD was chosen.

The server was up and running during the whole time for students to submit their code, for a period of one month and a half. CPU usage was kept below 15% on average and RAM usage was kept between 60% and 80% percent. A great percentage of this usage is due to the Docker containers and not to the traffic. Therefore, a dozen of additional users would have minimal impact on resource consumption.

4.5 Chapter Summary

This section describes in detail the development and implementation of COJ. From the technological perspective, it starts by eliciting system requirements, illustrating the architecture, and explaining how modules communicate. From the user's perspective, it

maps the user journey of the platform and how to perform some actions. Lastly, it gives a full picture of its infrastructure and how it is served to end-users.

5 Results and Discussion

This chapter presents the results of the surveys and the data collected from COJ. There were two cycles of experimentation in 2021 in a course that is destined for freshmen and introduces computer programming. The first cycle occurred from February until May with one class that is referred to as the class "2020.2 (14A)". The second cycle occurred from July until November with two classes that are referred to as "2021.1 (CC)" and "2021.1 (DD)". This course is generally taught in person, but due to the COVID-19 pandemic, students had remote classes from the beginning to the end. The following sections detail results from surveys and data analytics.

5.1 Surveys

All the students enrolled in these classes had the opportunity to respond to the three surveys. However, they were not obligated to do it because when responding by obligation answers may not be sincere. Therefore, the number of participants in each survey may change. Another determining factor is the possibility of dropping out of the class, hence the number of students at the end of the course may not be the same number that enrolled. Table 5 shows how many students from each class responded to surveys.

Table 5 – Amount of responses to the surveys

		Classes			Total
		2020.2 (14A)	2021.1 (CC)	2021.1 (DD)	
Students		59	79	66	204
Surveys	First	74.6%	98.7%	81.8%	86.3%
	Second	59.3%	58.2%	36.4%	51.6%
	Third	49.2%	38.0%	37.9%	41.2%

Source: the author

Each survey is applied at a different time during the semester because questions are related to the time they are in the course. Students were not directly identified but were required an email to respond to the survey in order to avoid duplicate answers. Questions descriptions and results are shown on Appendix D. Below there is a description of the purpose of each survey and what period they are shared with students.

- **First Survey:** It is applied at the beginning of the semester when students did not have any contact with the methodology yet. Approaches students' profiles and

expectations towards the course. These questions are important to certify if students are willing to dedicate themselves to the methodology by inquiring about their interest in trying it, which is a condition for obtaining relevant results in this experiment.

- **Second Survey:** It is applied right after finishing with Calango and going into C language. It intends to evaluate the user experience of both Calango and COJ with questions about their impressions and the SUS questionnaire (Questions 1 and 3). It gathers students' opinions about Calango before going to C.
- **Third Survey:** It is applied at the end of the semester when the term is completed. It has questions of the methodology as a whole and gathers students' views after going through C language and the URI online judge.

5.2 First Survey Results

The majority of the respondents are pursuing Software Engineering, and they represent 30.1% of the total and other Engineering degrees make up together 47.7%. The rest of the students are from other degrees or have not decided their major yet. Class 2020.2 (14A) is formed mostly by Software Engineering undergraduates. Class 2021.1 (CC) is formed mostly by Aerospace Engineering undergraduates, undecided students, and Software Engineering undergraduates. Lastly, Class 2021.1 (DD) is formed mostly by Software and Aerospace Engineering undergraduates.

None of the respondents consider computer programming NOT relevant for their professional life, and the majority consider it VERY relevant (68.2%), which affects the experiment positively by assuming that if they consider it important, they will put an effort to learn. Students that assigned the lowest relevance (slightly) from responses are all from Class 2020.2 (14A).

First semester students represent the majority being 56.3% of the participants, while 17.6% are in the 5th semester or beyond. Class 2021.1 (CC) has the highest percentage of first-semester students. The following questions reflect this distribution because Class 2021.1 (CC) has the least percentage of students that have been in contact with programming languages before this course. However, as a single group, most of the students have been in contact with programming languages somehow.

Lastly, more than half of the students have not experienced Calango or an online judge of any kind, but they all presented great interest in having this experience. Merely 7.4% did not want to use Calango or did not think it is necessary and 5.7% thought there were no benefits for them in using an online judge or did not want to experience it.

5.3 Second Survey

In relation to questions about Calango, students demonstrated enthusiasm with the tool, and besides one unsatisfied student and the indifferent responses (11.4%), all respondents were satisfied or very satisfied in using Calango as a programming tool. By reading the open responses of Question 12, we believe that the indifference comes from students that already know how to program, but did not feel harmed by Calango and understood the importance for their novice colleagues. One of these responses is transcribed below.

"[...] As I already have a knowledge of programming, I found it boring to use Calango, but I believe that for beginners it must have been very useful."

Regarding students that were satisfied but not very satisfied, all the suggestions for improvement in Calango are related to the help tab, which works as a manual and is accessible through the IDE. From one experiment to the other, some of these suggestions were addressed and implemented. These suggestions are transcribed below.

"As a suggestion, I think it would be in everyone's interest to have an improved Calango help tab [...]"

"I think it would be important to have a tab on the help page dedicated to the number of decimal places and their formatting in Calango."

"Improvement in the examples in the help area of calango, especially in the part of conditionals and repetitions."

"t would be ideal if the help box from Calango was complemented with some information, such as the possibility of choosing the number of decimal places of a number [...]"

Questions addressing COJ had three subjects: overall satisfaction, adequacy of the content, and new features. Although most of the students were satisfied (48.6%) or very satisfied (29.5%) using COJ, there were more unsatisfied students with it than with Calango, although it is a small percentage.

By reading open-ended responses and doing qualitative analysis, we believe that this slightly higher dissatisfaction is more related to the nature of both applications than with the quality of these platforms. While Calango is only related to their individual learning, COJ is used as an evaluative platform. That is, any problem with Calango IDE does not affect their final grade, while difficulties with COJ may affect their lists' conclusions and their grade. Therefore, most of the students' complaints on open-ended responses are related to difficulties in getting an accepted submission. Some of these complaints are transcribed below.

"[...] I received wrong answer several times, in which I did 34 tests and all correct, and even with help from colleagues and monitors I analyzed my program, and without being able to identify the error I gave up submitting [...]"

"Certain codes output correct results in all ways but were not accepted by COJ [...]"

"[...] better specify the problem that is preventing the algorithm from working, pointing out precisely the problem [...]"

"[...] it had some errors and I couldn't identify the error by myself [...]"

"[...] I think it would be easier to understand if it showed where the error is."

"COJ should have optional features to show where the programmer's error is [...]"

During the experiment, both the professor and the developer watched students closely and addressed as fast as possible any possible problems with COJ, especially because it is evaluating students. In the first list, there were a lot of students complaining that their code was correct and they did not receive an acceptance. By analyzing their submission using their ID, it was possible to see that students were missing requirements from the problem, which was a recurrent problem. They had difficulties understanding that if they did not strictly follow orientations from the problem, they would not get an accepted submission.

In relation to the adequacy of content, Table 6 shows a summary of students' positive responses from the survey. These questions are important to verify if the environment and problems were coherent with the course, and would not be a negative factor in the overall performance of the judge. For example, if students had been unsatisfied with COJ but marked that problems were too difficult, their dissatisfaction would not be concentrated on the tool itself, but on the quality of the problems.

Some of the negative responses about the adequacy of content were justified in the open field of Question 12. One problem of online judges is the inflexibility of output responses and some students were frustrated to keep fixing a functionally correct code but with wrong output formatting. Some suggested a more informative output, but this is controversial because results too informative keep students from thinking by themselves.

We also seized the opportunity to collect students' opinions on feature ideas, such

Table 6 – Percentage of students that selected positive impressions about COJ’s attributes in survey questions.

Impression about COJ	Percentage of students
Judgment was fair or very fair	78.1%
Problems’ difficulty level was adequate	70.5%
Problems’ descriptions were adequate, well written, or very well written	97.1%
Number of problems per list was balanced	87.6%

as theoretical questions and non-evaluative problems. The former was approved by approximately half of the students but it was discarded by us because we analyzed that the professor already disposed of other mature tools for theoretical questions, such as Moodle, which turned it less of a priority. Regarding non-evaluative problems, students showed a higher interest in them, therefore we developed it for the second cycle, and although it did not have lots of questions available, it was a well-received functionality.

Students’ responses also revealed that charts do have an impact on them. Therefore, we added more charts for them in the second cycle of experimentation. Moreover, the professor also has a dedicated page for seeing classes statistics besides the home page, which presented itself as a necessity for him after the first cycle.

Apart from the questions discussed above, we applied SUS questionnaires for Calango and COJ and obtained scores are listed in Table 5.3. According to [Bangor, Kortum and Miller \(2008\)](#), products should be above the 70 marks to be considered acceptable, but do not guarantee high acceptability in the field. Considering the results obtained and their confidence interval, Calango and COJ can be considered acceptable. From the first cycle to the second one, there were improvements in navigation and information available for users.

Table 7 – SUS scores obtained from survey

Tool	Class	Average	Standard deviation	Standard error	Confidence interval (95%)
Calango	2020.2 (14A)	75.1	16.8	2.8	± 5.6
	2021.1 (CC)	78.1	16.0	2.4	± 4.6
	2021.1 (DD)	83,5	13.2	2.7	± 5.3
COJ	2020.2 (14A)	73.7	21.9	3.7	± 7.3
	2021.1 (CC)	75.8	19.9	2.9	± 5.8
	2021.1 (DD)	79.1	22.2	4.5	± 8.9

Source: the author

5.4 Third Survey Results

One interesting result of this survey is that 84.5% of the students think that Calango made their learning easier, however, only 57.1% of the students attributed this event to the fact that Calango is in Portuguese. At first, one may think this refutes the importance of teaching novices in their native language but there is one other question that explains this distribution of opinions and we discuss this possibility below.

We questioned students about their English knowledge, and only 19.0% affirmed that they had basic knowledge, whereas the majority affirmed that they have advanced knowledge in English. Therefore, it makes sense that these students did not see Portuguese as a decisive factor to facilitate their learning, although it is not a hindrance. We believe that the students that have knowledge in English and are part of the 84.5% that agreed that Calango facilitates learning were more impacted by Calango's simplicity, pseudocode nature, and proximity to C language which prepared them for the next course activities.

This survey also questions students about the course's workload, rhythm, content coverage, and overall impression of the class, which all received positive feedback from the majority. Regarding online judges, 88.1% think that they reinforce their practical abilities and 75.0% think that their code quality was promoted by online judges. However, one particular result is concerning. When asked about which methodology makes them more motivated, only 65.5% of the students chose the online judge over the professor's manual testing.

If we go back to the previous survey and read open-ended responses, most of the students' frustration is turned towards receiving many unaccepted submissions and not finding the error. COJ and URI are not different from other online judges and this result may indicate that future works should focus on how to keep students motivated while using an online judge and how to improve their abilities in finding problems in their code without someone explicitly telling them.

Going to open-ended responses, we can highlight the feedback of a student that had taken this course without Calango and a student that suggested an improvement to the methodology that was incorporated in the following classes of 2021.1. They are transcribed below.

"This was the second time I took the discipline. The method that the professor approached this semester really made me learn! [...]"

"[Apply a] Similar project to that done as a final project in C but in calango before the content transition to C would help to scale larger programs."

5.5 Data Analytics

During the experiment, COJ received 9407 submissions from the three classes and general student results are listed in Table 8. The professor applied 5 lists with 5 evaluative problems each, and it was necessary to solve 75% of the total 25 evaluative problems as a condition to pass this course. Non-evaluative problems were available only at the second cycle because it was a feature derived from the first cycle.

Table 8 – General students results from COJ

Students that...	2020.2 (14A)	2021.1 (CC)	2021.1 (DD)	Total
enrolled in the course	59	79	66	204
submitted at least one source code to EP	79.7%	94.9%	95.4%	90.7%
solved at least one EP	78.0%	94.9%	92.4%	89.2%
solved 100% of EPs ($x=25$)	25.4%	54.4%	33.3%	39.2%
solved between 75% and 99% of EPs ($19 \leq x < 25$)	25.4%	19.0%	43.9%	28.9%
solved between 50% and 74% of EPs ($13 \leq x < 19$)	11.9%	16.5%	6.1%	11.8%
solved between 25% and 49% of EPs ($6 \leq x < 13$)	8.5%	2.5%	6.1%	5.4%
solved between 1% and 25% of EPs ($1 \leq x < 6$)	6.8%	2.5%	3.0%	3.9%
solved all EPs tried	63.8%	80.0%	66.7%	64.7%
tried at least one NEP	-	25.3%	21.2%	*23.4%
solved at least one NEP	-	25.3%	19.7%	*22.8%

*percentage considering only classes CC and DD

EP=Evaluative problem; NEP=Non-evaluative problem

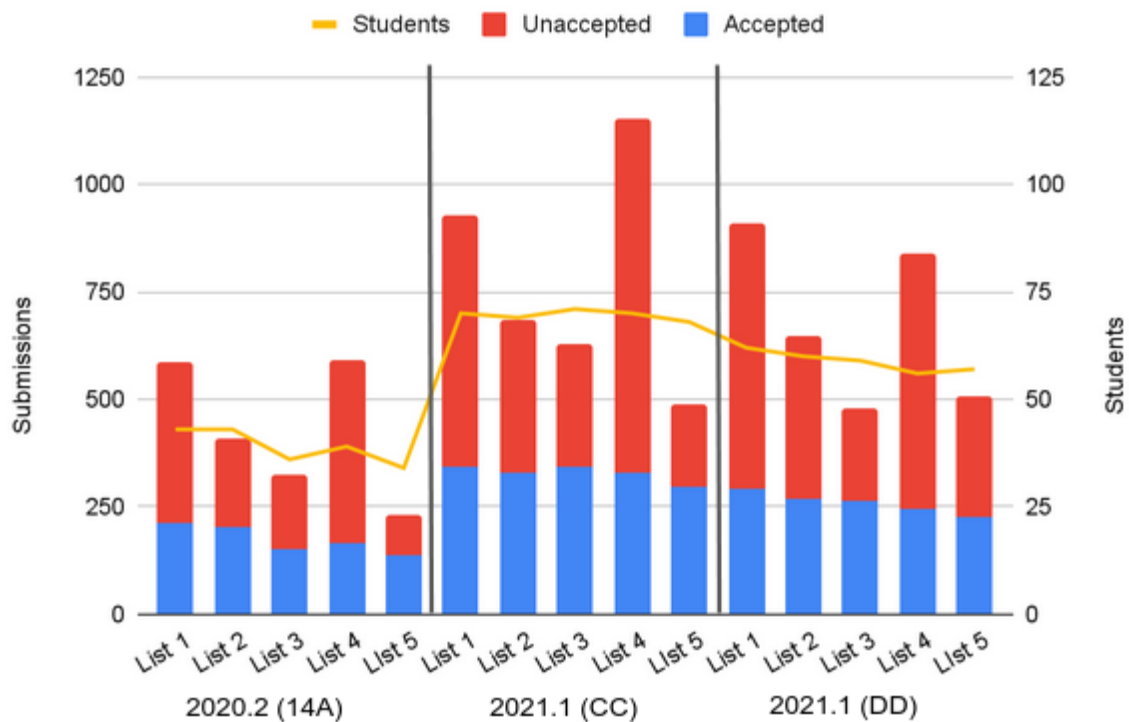
Source: the author

With regard to submission statistics, Figure 17 shows the number of submissions that were accepted or not and Figure 18 shows detailed information about unaccepted submissions in each list. The most common error was WRONG ANSWER. This may be alarming at first but it is not different from the results of other online judges (RUBIO-SÁNCHEZ et al., 2014; MANZOOR, 2006).

It is possible to see in Figure 17 that unaccepted submissions start high in the first list and drop as students progress list by list, but in List 4 there is an increase in this average. The high amount of submissions in the first list is normal since students are getting to know the online judge and how it works. However, the high amount in the fourth list is peculiar. Looking deeper into the data, we found that particularly 2 questions from list 4 raised this average: a Fibonacci problem and a date validator problem.

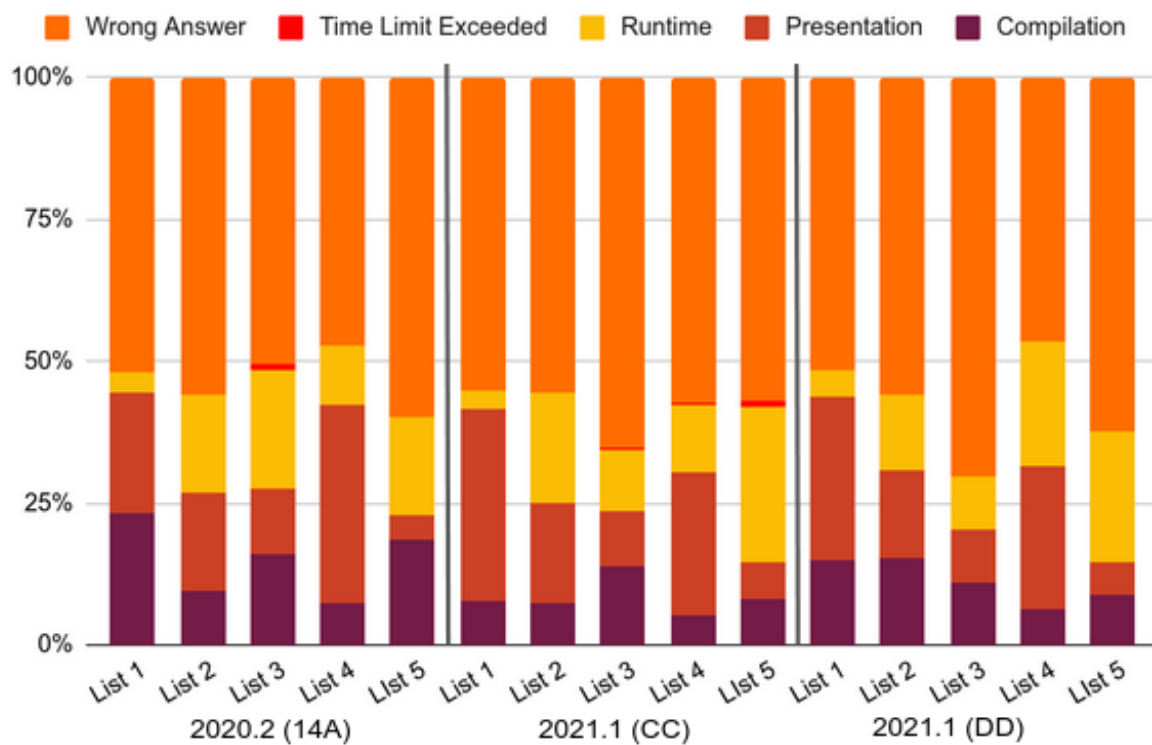
The Fibonacci problem raised the PRESENTATION ERROR counting because it was the first question to request inline outputs and most of the students wrongly printed a space at the end of their output. The date validator was one of the problems with many extreme cases and this raised the WRONG ANSWER counting because it

Figure 17 – Number of accepted and unaccepted submissions per list



Source: the author

Figure 18 – Distribution of errors of unaccepted submissions per list

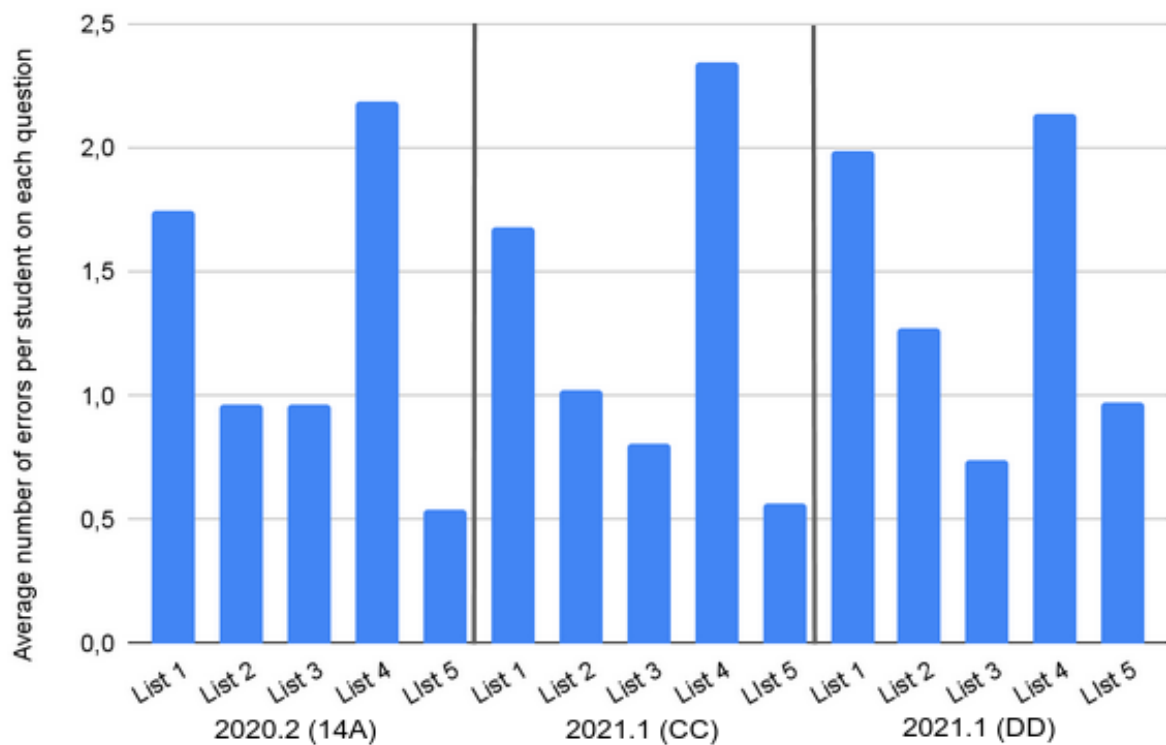


Source: the author

required students to think through all the possibilities and many were not covering all cases, especially the hidden ones.

Apart from this raise in List 4, the decrease in the average of unaccepted submissions from students is an indicator of progress. Of course, COJ requires a piece of knowledge on how to operate online judges, such as knowing how to print outputs accordingly, but we can also infer that students progressed in their problem-solving ability, since questions got more difficult but their error rate decreased. To reinforce this hypothesis, Figure 19 shows the error rate by student in each question. It shows that students have around 2 errors until they get an accepted submission and, in the last list, the average is under 1, which implies that most of students got an accepted submission on their first attempt.

Figure 19 – Average number of unaccepted submissions per student on each question of a list.



Source: the author

Another interesting aspect to analyze is how students dealt with deadlines. Table 9 details the periods in which the lists were open for each class. Initially, every list was supposed to be opened for eight days. System maintenance was responsible for leaving only seven days for one of the lists. Regarding lists with nine days, the professor provided extensions to attend to students' requests, but these lists were scheduled with an eight-day duration at first.

Students' submissions over time may indicate success in solving all problems. Let us consider a successful result as the student solving all 25 evaluative problems available,

Table 9 – Deadlines of each list applied during the experiment

Class	List	Start	End	Days
2020.2 (14A)	1	05/02/2021 10:00	12/02/2021 23:59	8
	2	13/02/2021 13:00	19/02/2021 23:59	7
	3	19/02/2021 14:00	26/02/2021 23:59	8
	4	26/02/2021 14:00	05/03/2021 23:59	8
	5	05/03/2021 14:00	12/03/2021 23:59	8
2021.1 (CC)	1	23/07/2021 08:00	30/07/2021 23:59	8
	2	30/07/2021 08:00	06/08/2021 23:59	8
	3	06/08/2021 08:00	13/08/2021 23:59	8
	4	13/08/2021 08:00	21/08/2021 23:59	9
	5	20/08/2021 08:00	28/08/2021 23:59	9
2021.1 (DD)	1	23/07/2021 08:00	30/07/2021 23:59	8
	2	30/07/2021 08:00	06/08/2021 23:59	8
	3	06/08/2021 08:00	13/08/2021 23:59	8
	4	13/08/2021 08:00	21/08/2021 23:59	9
	5	20/08/2021 08:00	28/08/2021 23:59	9

Source: the author

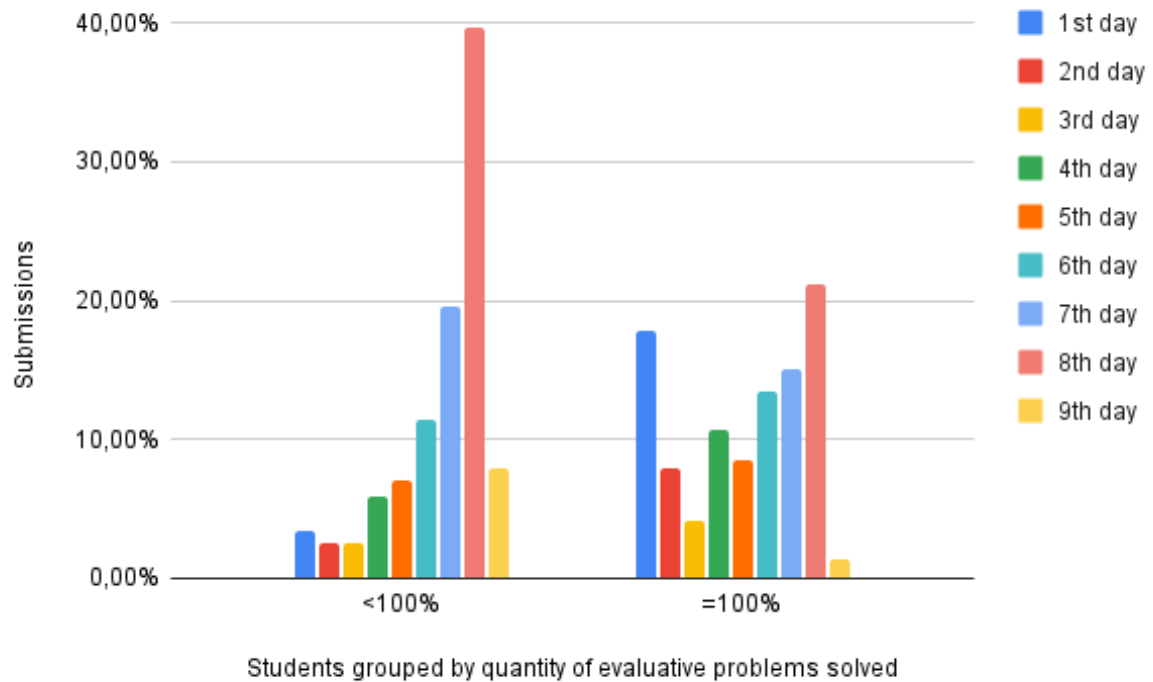
and compare them with students who did not solve 100% of the problems. Figure 20 shows that the group who completed all problems have worked on them along all week, and their percentage amount of submissions is well distributed across all days, while the other group has a higher difference in percentages between the farthest and closest days to deadlines.

We only compared students from the experiment with themselves to detect behavior that can lead to success, that is, solving all problems. However, to validate COJ as an effective methodology, it is necessary to compare it with a control group. Considering that after learning Calango and using COJ, students start learning C and using the URI online judge, it is possible to compare URI results from the experimental group with a control group.

Table 10 list four 2019 classes that will compose the control group. They had the same course, professor, and syllabus, but they did not use COJ. Another difference is that the control group had in-person classes, and the experimental group did not. The four URI problems listed by their identifier were applied to each class and the number of average attempts per student is provided by URI.

If we aggregate results from the control and experimental groups, the experimental group needed fewer attempts in three of the four problems. Still, we see that there is a visible difference between classes. Although in surveys students' responses for each class are very close, the data analysis shows that classes are different from one another and it turns out their groups of students were not as random as expected. This difference was also observed in the action research. Class 2020.2(14A) was attentive but less participating,

Figure 20 – Students’ submissions per day of list grouped by problems solved



Source: the author

Table 10 – Students’ average attempts until acceptance per URI problem

		Control classes				Experimental classes		
		2019.1 (DD)	2019.1 (EE)	2019.2 (DD)	2019.2 (EE)	2020.2 (14A)	2021.1 (CC)	2021.1 (DD)
Problems	#1008	2.9	3.2	4.2	2.8	1.9	2.4	3.1
	#1015	1.9	2.9	3.0	2.3	2.0	2.9	2.8
	#1019	1.8	2.4	2.1	2.1	1.7	1.8	2.1
	#1115	2.1	1.8	2.4	2.3	1.7	2.2	2.1
	Average	2.2	2.6	2.9	2.4	1.8	2.3	2.5

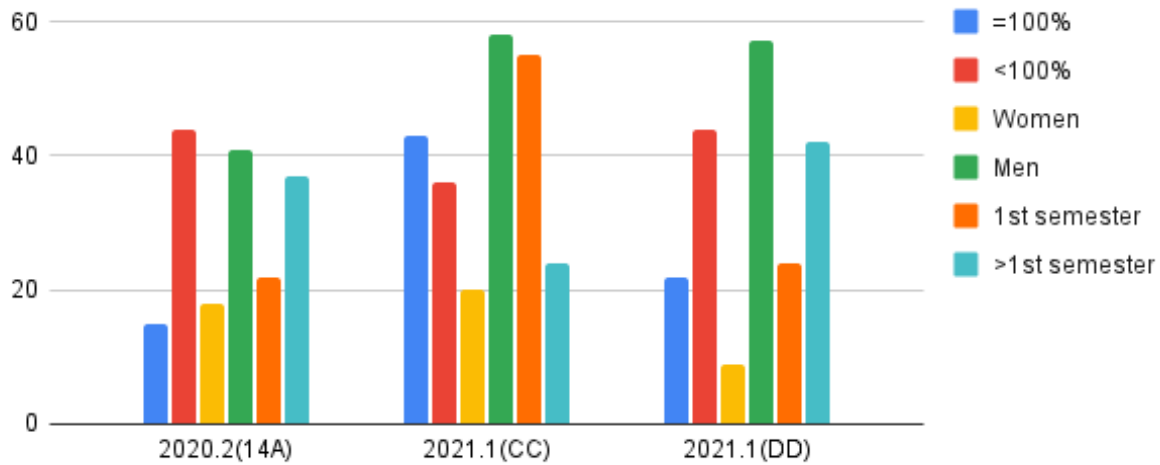
Source: the author

class CC was proactive but less efficient, and class DD was less participating and less efficient.

Going back to COJ’s data, Figure 21 separates students per class and by the following criteria: percentage of the conclusion of lists, freshmen or veterans, and gender. Proportions by gender seem to be regular among classes, but lists conclusion and number of freshmen differ in class CC. Interestingly, they differ proportionally, that is, the higher the number of freshmen, the higher the number of students who conclude 100% of the lists.

This directly proportional difference takes us to a second analysis, in which we check the percentage of students that concluded 100% of lists and are also freshmen.

Figure 21 – Number of students per class grouped by list conclusion, gender, and semester



Source: the author

Results show that 55,45% of 101 freshmen students (1st semester) concluded 100% of EPs, whereas only 23,30% of non-freshmen (>1st semester) concluded 100% of EPs. This makes sense if we hypothesize that many of the veterans of this course are probably working as well and therefore having less time to dedicate to the course. Regarding veterans from the 5th semester and beyond, they are probably in courses other than software engineering that do not have programming as a direct requirement, which makes them not dedicate themselves as much as software engineering students.

Another analysis from a different perspective shows that veterans do not have more difficulties than freshmen to conclude EPs. Table 11 shows average attempts of concluded EPs per student for crossed groups. Independently of list conclusion percentage, veterans have similar averages to conclude EPs, whereas freshmen that did not conclude 100% of EPs (row=<100% & column=1st semester) have the highest average of them all (3.08). We can infer that veterans that do not conclude 100% of EPs are probably more affected by the time they have available to dedicate to the course than difficulties in solving problems.

Table 11 – Average attempts per student for each concluded EP grouped by list conclusion, gender, and semester

	Men	Women		
1st semester	2.52	2.71		
>1st semester	2.16	2.31	1st semester	>1st semester
<100%	2.63	2.22	3.08	2.27
=100%	2.17	2.51	2.28	2.10

Source: the author

5.6 Threats to Validity

This work may suffer interference from many factors attached to the environment of experimentation or the survey instruments. Although it does not diminish the contribution of this work to programming education, it is important to address its limitations. Therefore, in this section, we discuss some variables that can impact our results.

Regarding the environment, experimental classes were conducted virtually during all time considering the COVID-19 pandemic, while control classes were conducted in person. Moreover, classes had different schedules although they all had three meetings (synchronous or asynchronous) per week with a 2-hour duration at most. Another interfering factor is that this class is for freshmen and each semester freshmen are picked out from different public exams (Enem or UnB), which makes our sample not so random and different from one another.

Other environmental conditions were identical or similar, but the differences noted above may affect student's behavior. For example, students having classes in the morning may be more attentive than in the afternoon (or vice-versa). Concerning the experimental group having virtual classes, it can be either beneficial or harmful.

About research instruments, there are some caveats inherent to unsupervised survey questionnaires, which are: students may not have been sincere due to fear of giving a negative answer; students may not pay attention to questions to finish the survey faster; students perception about a teaching methodology may not assess well their actual learning (DESLAURIERS et al., 2019).

To mitigate interferences in the research instruments: we added data analytics as another layer of investigation; we assured students that they would not be identified in surveys; we did not obligate students to answer the survey as part of class activities to avoid uninterested responses. Lastly, qualitative results may suffer biases from researchers and their personal experience, therefore we suggest to readers to interpret qualitative results having in mind all the quantitative pieces of evidence as well.

5.7 Chapter Summary

This section presents the quantitative and qualitative results from COJ. It starts by discussing the main results from surveys and data analytics. Overall, students had positive opinions about this methodology. In relation to their behavior, data show a tendency of them submitting code closer to the due date. Lastly, this section addresses threats to validity by discussing interfering variables, such as the pandemic, classes' different schedules, and biases from students and researchers.

6 Conclusion

This work approached the development of the Calango Online Judge (COJ) and the first cycle of experimentation on an undergraduate course class. The experiment was conducted through two cycles of action research, in which we applied surveys. We also seized the opportunity to question students about Calango as well. Between cycles, there were improvements in the software. Our main findings were:

- Most students had a satisfactory experience with COJ and Calango as programming tools.
- The main problem with online judges is how to keep students motivated to keep trying when they get a lot of unaccepted submissions and do not find their errors.
- Students concentrate their submissions closer to the deadline. However, students that solved all problems present a better distribution of submissions along all weekdays.
- Comparing the same problems of URI Online Judge across control groups and the experimental group, we found that, in general, students that used Calango with COJ needed fewer attempts to solve these problems than students that did not use COJ while learning with Calango.
- Classes are not as random as expected. Each class has a different characteristic that may work better with different approaches.

In summary, our main contributions to the area are that we found that teaching non-English speaking novices in their native language is significantly important for their learning, whereas English-speaking students benefit most from simpler languages before going into more complex languages. Also, online judges improve their abilities but professors should monitor their motivation and frustration while getting their code automatically tested, besides analyzing the frequency of their submission along weekdays to predict their final result. Future works may include the addition of new features to COJ and other experiments comparing the usage of Calango and COJ with classes that learn C language without them.

References

- ACM Library. Association for Computing Machinery, 2021. Available at: <https://dl.acm.org>. Accessed on: August 14, 2021. Cited on page 41.
- ALA-MUTKA, K. M. A survey of automated assessment approaches for programming assignments. *Computer science education*, Taylor & Francis, v. 15, n. 2, p. 83–102, 2005. Cited on page 36.
- ALEKSIĆ, V.; IVANOVIĆ, M. Introductory programming subject in european higher education. *Informatics in Education*, Vilnius University Institute of Data Science and Digital Technologies, v. 15, n. 2, p. 163–182, 2016. Cited 2 times on pages 30 and 33.
- ALEMAN, J. L. F. Automated assessment in a programming tools course. *IEEE Transactions on Education*, v. 54, n. 4, p. 576–581, 2011. Cited on page 35.
- AUSUBEL, D.; NOVAK, J.; HANESIAN, H. *Educational Psychology: A Cognitive View*. Holt, Rinehart and Winston, 1978. ISBN 9780030899515. Available at: <https://books.google.com.br/books?id=17cdAAAAMAAJ>. Cited on page 29.
- BANGOR, A.; KORTUM, P. T.; MILLER, J. T. An empirical evaluation of the system usability scale. *International Journal of Human–Computer Interaction*, Taylor & Francis, v. 24, n. 6, p. 574–594, 2008. Available at: <https://doi.org/10.1080/10447310802205776>. Cited 2 times on pages 41 and 69.
- BEYNON-DAVIES, P. et al. Rapid application development (rad): an empirical review. *European Journal of Information Systems*, Taylor & Francis, v. 8, n. 3, p. 211–223, 1999. Cited on page 41.
- BEZ, J. L.; TONIN, N. A.; RODEGHERI, P. R. Uri online judge academic: A tool for algorithms and programming classes. In: *2014 9th International Conference on Computer Science Education*. [S.l.: s.n.], 2014. p. 149–152. Cited on page 35.
- BROOKE, J. Sus: A 'quick and dirty' usability scale. In: *Usability Evaluation In Industry*. 1st edition. ed. London: Taylor & Francis, 1996. chap. 12, p. 189–194. Cited 2 times on pages 40 and 125.
- CANEDO, E. D.; SANTOS, G. A.; FREITAS, S. A. A. de. Analysis of the teaching-learning methodology adopted in the introduction to computer science classes. In: *IEEE. 2017 IEEE Frontiers in Education Conference (FIE)*. [S.l.], 2017. p. 1–8. Cited on page 25.
- CANEDO, E. D.; SANTOS, G. A.; LEITE, L. L. An assessment of the teaching-learning methodologies used in the introductory programming courses at a brazilian university. *Informatics in Education*, Vilnius University Institute of Mathematics and Informatics, v. 17, n. 1, p. 45–59, 2018. Cited 3 times on pages 26, 30, and 33.
- CodeChef. Sphere Research Labs., 2021. Available at: <https://www.codechef.com>. Accessed on: August 12, 2021. Cited on page 34.

- Codeforces. Mike Mirzayanov, 2021. Available at: <https://codeforces.com>. Accessed on: August 12, 2021. Cited on page 34.
- COLLATTO, D. et al. Is action design research indeed necessary? analysis and synergies between action research and design science research. *Systemic Practice and Action Research*, v. 31, 06 2018. Cited on page 40.
- DESLAURIERS, L. et al. Measuring actual learning versus feeling of learning in response to being actively engaged in the classroom. *Proceedings of the National Academy of Sciences*, v. 116, p. 201821936, 09 2019. Cited on page 77.
- DigitalOcean. DigitalOcean LLC, 2021. Available at: <https://www.digitalocean.com>. Accessed on: August 14, 2021. Cited on page 44.
- Django Framework (Version 3.1.4). Django Software Foundation, 2020. Available at: <https://docs.djangoproject.com/en/3.1>. Accessed on: August 14, 2021. Cited on page 44.
- Docker. Docker Inc, 2021. Available at: <https://www.docker.com>. Accessed on: August 14, 2021. Cited on page 44.
- DOUCE, C.; LIVINGSTONE, D.; ORWELL, J. Automatic test-based assessment of programming: A review. *Journal on Educational Resources in Computing (JERIC)*, ACM New York, NY, USA, v. 5, n. 3, p. 4–es, 2005. Cited on page 34.
- DREW, M. R.; FALCONE, B.; BACCUS, W. L. What does the system usability scale (sus) measure? In: MARCUS, A.; WANG, W. (Ed.). *Design, User Experience, and Usability: Theory and Practice*. Cham: Springer International Publishing, 2018. p. 356–366. ISBN 978-3-319-91797-9. Cited on page 40.
- EASTERBROOK, S. et al. Selecting empirical methods for software engineering research. In: _____. *Guide to Advanced Empirical Software Engineering*. London: Springer London, 2008. p. 285–311. ISBN 978-1-84800-044-5. Available at: https://doi.org/10.1007/978-1-84800-044-5_11. Cited on page 40.
- EZENWOYE, O. What language? - the choice of an introductory programming language. In: *2018 IEEE Frontiers in Education Conference (FIE)*. Los Alamitos, CA, USA: IEEE Computer Society, 2018. p. 1–8. Available at: <https://doi.ieeecomputersociety.org/10.1109/FIE.2018.8658592>. Cited on page 33.
- FEDORENKO, E. et al. The language of programming: A cognitive perspective. *Trends in Cognitive Sciences*, v. 23, n. 7, p. 525–528, 2019. ISSN 1364-6613. Available at: <https://www.sciencedirect.com/science/article/pii/S1364661319301020>. Cited on page 29.
- FELINTO, C.; GIROTTTO, V. *Projeto Calango - Editor e Interpretador de Algoritmos*. Bachelor's Thesis — Universidade Católica de Brasília (UCB), 2012. Cited 2 times on pages 30 and 31.
- FORSYTHE, G. E.; WIRTH, N. Automatic grading programs. *Communications of the ACM*, ACM New York, NY, USA, v. 8, n. 5, p. 275–278, 1965. Cited on page 34.
- Git. Software Freedom Conservancy, 2021. Available at: <https://git-scm.com>. Accessed on: August 14, 2021. Cited on page 44.

- GitHub. GitHub Inc., 2021. Available at: <<https://github.com>>. Accessed on: August 14, 2021. Cited on page 44.
- Google Forms. Google LLC, 2021. Available at: <<https://www.google.com/intl/pt-BR/forms/about>>. Accessed on: August 14, 2021. Cited on page 41.
- Google Scholar. Google LLC, 2021. Available at: <<https://scholar.google.com>>. Accessed on: August 14, 2021. Cited on page 41.
- GOUIGOUX, J.; TAMZALIT, D. From monolith to microservices: Lessons learned on an industrial migration to a web oriented architecture. In: *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. [S.l.: s.n.], 2017. p. 62–65. Cited on page 61.
- GUO, P. J. Non-native english speakers learning computer programming: Barriers, desires, and design opportunities. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2018. (CHI '18), p. 1–14. ISBN 9781450356206. Available at: <<https://doi.org/10.1145/3173574.3173970>>. Cited on page 29.
- HALEY, D. T. et al. Seeing the whole picture: Evaluating automated assessment systems. *Innovation in Teaching and Learning in Information and Computer Sciences*, Routledge, v. 6, n. 4, p. 203–224, 2007. Available at: <<https://doi.org/10.11120/ital.2007.06040203>>. Cited on page 36.
- IEEE Xplore. IEEE, 2021. Available at: <<https://ieeexplore.ieee.org>>. Accessed on: August 14, 2021. Cited on page 41.
- IHANTOLA, P. et al. Review of recent systems for automatic assessment of programming assignments. In: *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*. New York, NY, USA: Association for Computing Machinery, 2010. (Koli Calling '10), p. 86–93. ISBN 9781450305204. Available at: <<https://doi.org/10.1145/1930464.1930480>>. Cited on page 37.
- ISO/IEC 25010. *ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. 2011. Cited on page 47.
- KHAN, A. et al. Unit-4 descriptive, experimental and action research. In: *Block-2 Research Methods for Distance Education*. [S.l.]: IGNOU, 2018. p. 67–96. Cited on page 40.
- KURNIA, A.; LIM, A.; CHEANG, B. Online judge. *Computers & Education*, v. 36, n. 4, p. 299–315, 2001. ISSN 0360-1315. Available at: <<https://www.sciencedirect.com/science/article/pii/S0360131501000185>>. Cited on page 33.
- LEWIS, J. R. The system usability scale: Past, present, and future. *International Journal of Human-Computer Interaction*, Taylor & Francis, v. 34, n. 7, p. 577–590, 2018. Available at: <<https://doi.org/10.1080/10447318.2018.1455307>>. Cited on page 40.
- LUCASSEN, G. et al. The use and effectiveness of user stories in practice. In: . [S.l.: s.n.], 2016. p. 205–222. ISBN 978-3-319-30281-2. Cited on page 47.

MALIK, S. I. Improvements in introductory programming course: action research insights and outcomes. *Systemic Practice and Action Research*, Springer, v. 31, n. 6, p. 637–656, 2018. Cited 2 times on pages 33 and 40.

MANZOOR, S. Analyzing programming contest statistics. *Perspectives on Computer Science Competitions for (High School) Students*, v. 48, 2006. Cited on page 71.

MARTIN, J. *Rapid Application Development*. Macmillan Publishing Company, 1991. (The James Martin productivity series). ISBN 9780023767753. Available at: <https://books.google.com.br/books?id=o6FQAAAAMAAJ>. Cited on page 41.

MARTINS, A. I. et al. European portuguese validation of the system usability scale (sus). *Procedia Computer Science*, v. 67, p. 293–300, 2015. ISSN 1877-0509. Proceedings of the 6th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion. Available at: <https://www.sciencedirect.com/science/article/pii/S1877050915031191>. Cited 2 times on pages 41 and 126.

Maven. The Apache Software Foundation, 2021. Available at: <https://maven.apache.org>. Accessed on: August 15, 2021. Cited 2 times on pages 44 and 51.

MEDEIROS, R. P.; RAMALHO, G. L.; FALCÃO, T. P. A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, v. 62, n. 2, p. 77–90, 2019. Cited on page 25.

NGINX (Version 1.19.0). F5 Inc, 2021. Available at: <https://www.nginx.com>. Accessed on: August 14, 2021. Cited on page 44.

PIETERSE, V. Automated assessment of programming assignments. In: *Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research*. Heerlen, NLD: Open Universiteit, Heerlen, 2013. (CSERC '13), p. 45–56. Cited on page 36.

PORTNOFF, S. R. The introductory computer programming course is first and foremost a *Language* course. *ACM Inroads*, Association for Computing Machinery, New York, NY, USA, v. 9, n. 2, p. 34–52, Apr. 2018. ISSN 2153-2184. Available at: <https://doi.org/10.1145/3152433>. Cited on page 29.

Portugol IDE 2.2. Instituto Politécnico de Tomar, 2006. Available at: <http://orion.ipt.pt/~manso/Portugol>. Accessed on: August 14, 2021. Cited 2 times on pages 31 and 33.

Portugol Studio. Alisson Steffens, 2021. Available at: <http://univali-lite.github.io/Portugol-Studio>. Accessed on: August 14, 2021. Cited 2 times on pages 31 and 33.

PostgreSQL (Version 13.0). The PostgreSQL Global Development Group, 2020. Available at: <https://www.postgresql.org/docs/13/release-13.html>. Accessed on: August 14, 2021. Cited on page 44.

REVILLA, M. A.; MANZOOR, S.; LIU, R. Competitive learning in informatics: The uva online judge experience. *Olympiads in Informatics*, Institute of Mathematics and Informatics, v. 2, n. 10, p. 131–148, 2008. Cited on page 34.

- ROUSSEL, S. et al. Learning subject content through a foreign language should not ignore human cognitive architecture: A cognitive load theory approach. *Learning and Instruction*, v. 52, p. 69–79, 2017. ISSN 0959-4752. Available at: <https://www.sciencedirect.com/science/article/pii/S0959475216302584>. Cited on page 29.
- RUBIO-SÁNCHEZ, M. et al. Student perception and usage of an automated programming assessment tool. *Computers in Human Behavior*, v. 31, p. 453–460, 2014. ISSN 0747-5632. Available at: <https://www.sciencedirect.com/science/article/pii/S0747563213001040>. Cited on page 71.
- SERRANO, N.; HERNANTES, J.; GALLARDO, G. Service-oriented architecture and legacy systems. *IEEE Software*, v. 31, n. 5, p. 15–19, 2014. Cited on page 50.
- SILVA, G. et al. Impact of calango language in an introductory computer programming course. In: *2020 IEEE Frontiers in Education Conference (FIE)*. [S.l.: s.n.], 2020. p. 1–9. Cited 5 times on pages 26, 30, 32, 33, and 39.
- SKALKA, J.; DRLÍK, M.; OBONYA, J. Automated assessment in learning and teaching programming languages using virtual learning environment. In: *2019 IEEE Global Engineering Education Conference (EDUCON)*. [S.l.: s.n.], 2019. p. 689–697. Cited on page 35.
- ŠPAČEK, F.; SOHLICH, R.; DULÍK, T. Docker as platform for assignments evaluation. *Procedia Engineering*, v. 100, p. 1665–1671, 2015. ISSN 1877-7058. 25th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2014. Available at: <https://www.sciencedirect.com/science/article/pii/S1877705815005688>. Cited on page 36.
- SPOJ. Sphere Research Labs., 2021. Available at: <https://spoj.com>. Accessed on: August 12, 2021. Cited on page 34.
- Spring Boot (Version 2.4.0). VMware Inc, 2020. Available at: <https://spring.io/>. Accessed on: August 14, 2021. Cited on page 44.
- Springer. Springer Nature Switzerland AG, 2021. Available at: <https://www.springer.com>. Accessed on: August 14, 2021. Cited on page 41.
- Taylor & Francis Online. Informa UK Limited, 2021. Available at: <https://www.tandfonline.com>. Accessed on: August 14, 2021. Cited on page 41.
- TONIN, N. A.; BEZ, J. L. Uri online judge: A new classroom tool for interactive learning. In: *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)*. [S.l.: s.n.], 2012. p. 1. Cited on page 35.
- ULLAH, Z. et al. The effect of automatic assessment on novice programming: Strengths and limitations of existing systems. *Computer Applications in Engineering Education*, v. 26, n. 6, p. 2328–2341, 2018. Available at: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cae.21974>. Cited on page 36.
- UVa. University of Valladolid, 2021. Available at: https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=23. Accessed on: August 12, 2021. Cited on page 34.

VEERASAMY, A. K.; SHILLABEER, A. Teaching english based programming courses to english language learners/non-native speakers of english. *International Proceedings of Economics Development and Research*, IACSIT Press, v. 70, p. 17, 2014. Cited on page 25.

VERDÚ, E. et al. A distributed system for learning programming on-line. *Computers & Education*, v. 58, n. 1, p. 1–10, 2012. ISSN 0360-1315. Available at: <https://www.sciencedirect.com/science/article/pii/S036013151100193X>. Cited on page 35.

Visualg 3.0. Emepplus, 2017. Available at: <https://visualg3.com.br>. Accessed on: August 14, 2021. Cited 2 times on pages 31 and 33.

WANG, G. P. et al. Ojpot: online judge & practice oriented teaching idea in programming courses. *European Journal of Engineering Education*, Taylor & Francis, v. 41, n. 3, p. 304–319, 2016. Available at: <https://doi.org/10.1080/03043797.2015.1056105>. Cited on page 35.

WANG, S. Research on the teaching strategies of senior high school english listening guided by the meaningful learning theory. *International Journal of Liberal Arts and Social Science*, v. 8, n. 10, 2020. Cited on page 29.

WASIK, S. et al. A survey on online judge systems and their applications. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 51, n. 1, Jan. 2018. ISSN 0360-0300. Available at: <https://doi.org/10.1145/3143560>. Cited 2 times on pages 34 and 36.

WATSON, C.; LI, F. W. Failure rates in introductory programming revisited. In: *Proceedings of the 2014 conference on Innovation & technology in computer science education*. [S.l.: s.n.], 2014. p. 39–44. Cited on page 25.

WILCOX, C. The role of automation in undergraduate computer science education. In: *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2015. (SIGCSE '15), p. 90–95. ISBN 9781450329668. Available at: <https://doi.org/10.1145/2676723.2677226>. Cited on page 35.

WU, H. et al. Online judge system and its applications in c language teaching. In: *2016 International Symposium on Educational Technology (ISET)*. [S.l.: s.n.], 2016. p. 57–60. Cited on page 35.

YI, C.; FENG, S.; GONG, Z. A comparison of sandbox technologies used in online judge systems. In: *Mechanical Design and Power Engineering*. [S.l.]: Trans Tech Publications Ltd, 2014. (Applied Mechanics and Materials, v. 490), p. 1201–1204. Cited on page 36.

ZOWGHI, D.; COULIN, C. Requirements elicitation: A survey of techniques, approaches, and tools. In: _____. *Engineering and Managing Software Requirements*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 19–46. Available at: https://doi.org/10.1007/3-540-28244-0_2. Cited on page 42.

Appendix

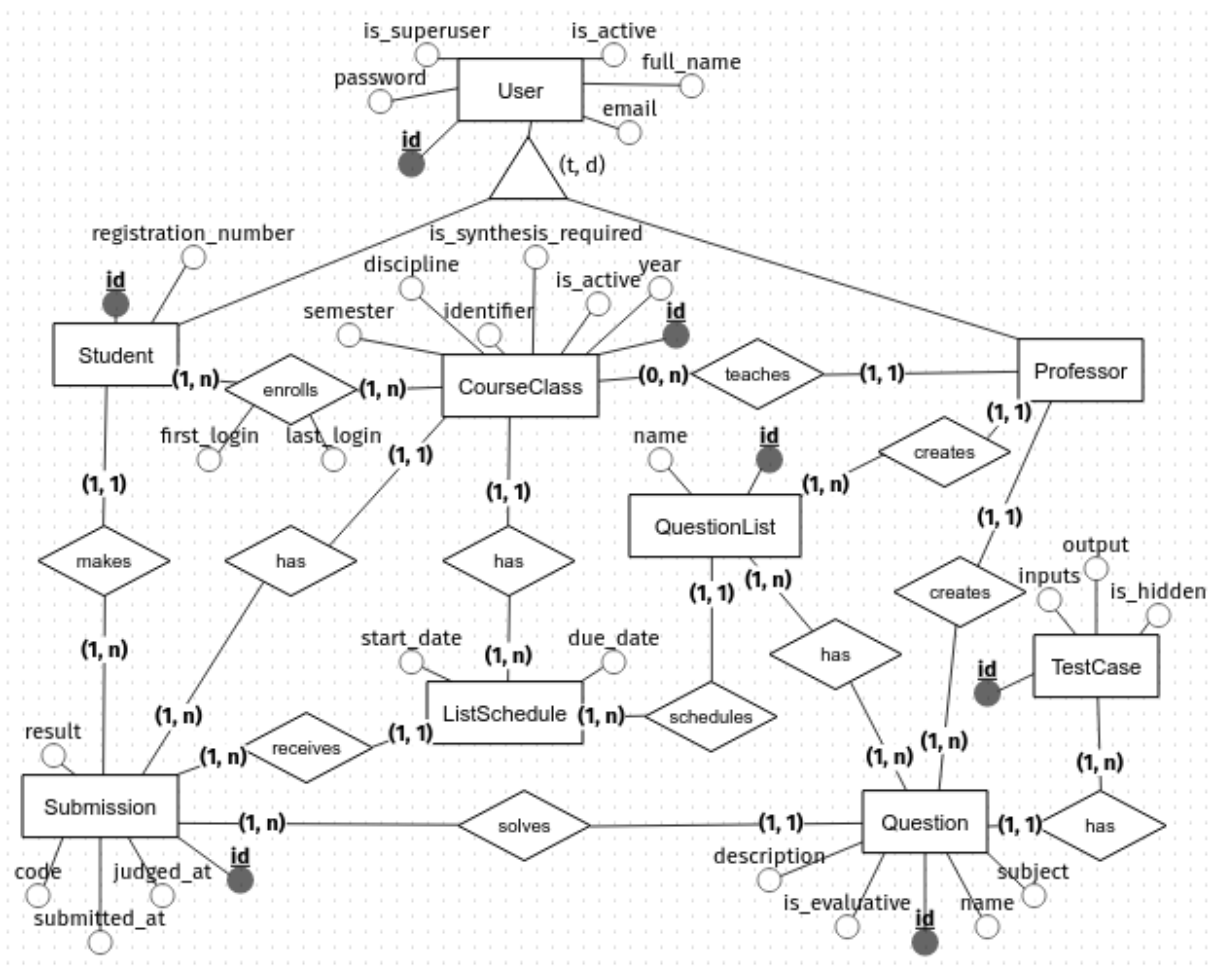
APPENDIX A – Source Code Repositories

All the source code used in this work is open at GitHub. See the list of repositories below:

- Calango Online Judge (COJ) (<https://github.com/GeovanaRamos/calango-online-judge>)
- Judge Microservice (<https://github.com/GeovanaRamos/calango-judge-service>)
- Calango Desktop App (<https://github.com/GeovanaRamos/calango>)
- Calango Interpreter Package (<https://github.com/GeovanaRamos/calango-interpreter>)

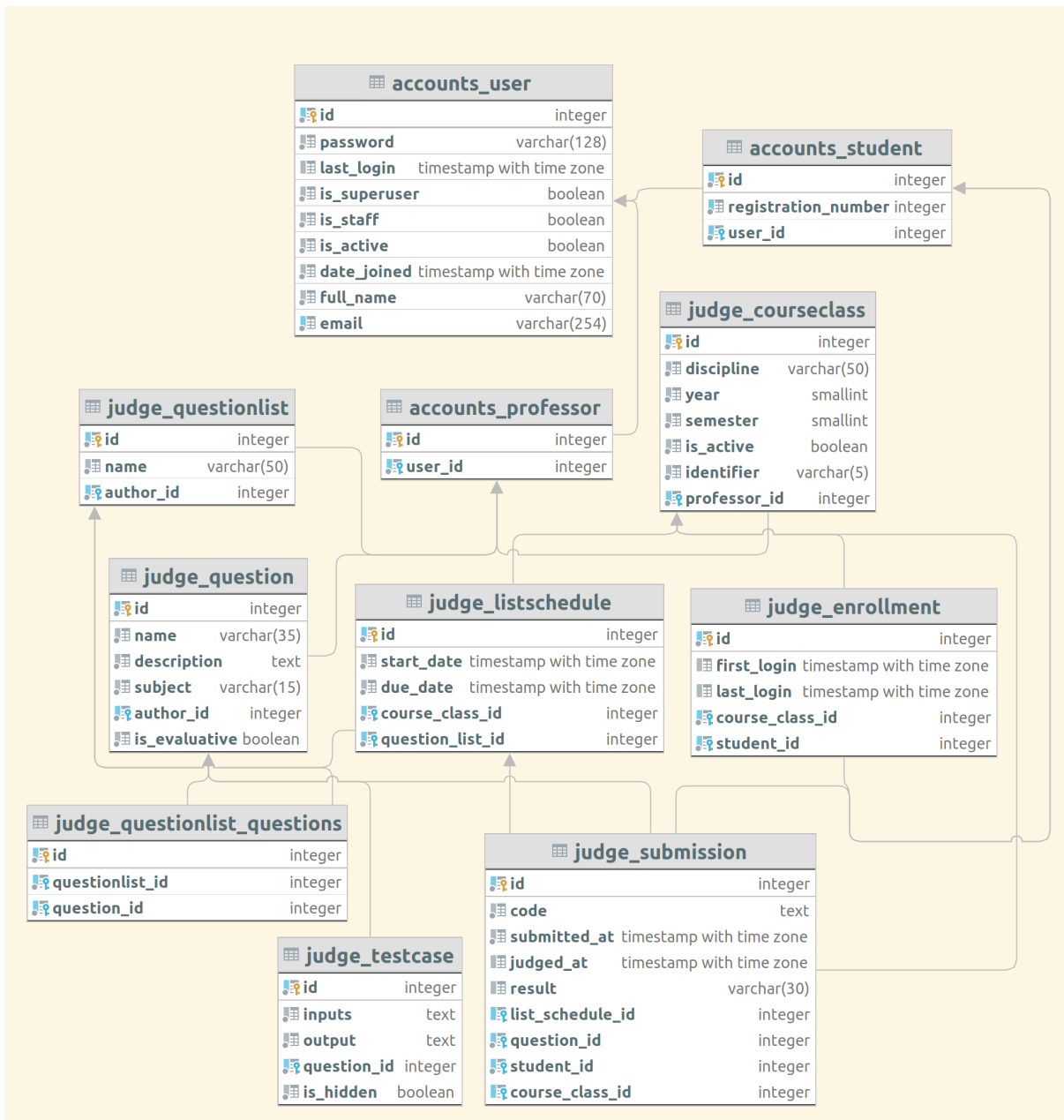
APPENDIX B – Artifacts

Figure 22 – Entity-relationship diagram of the Django application



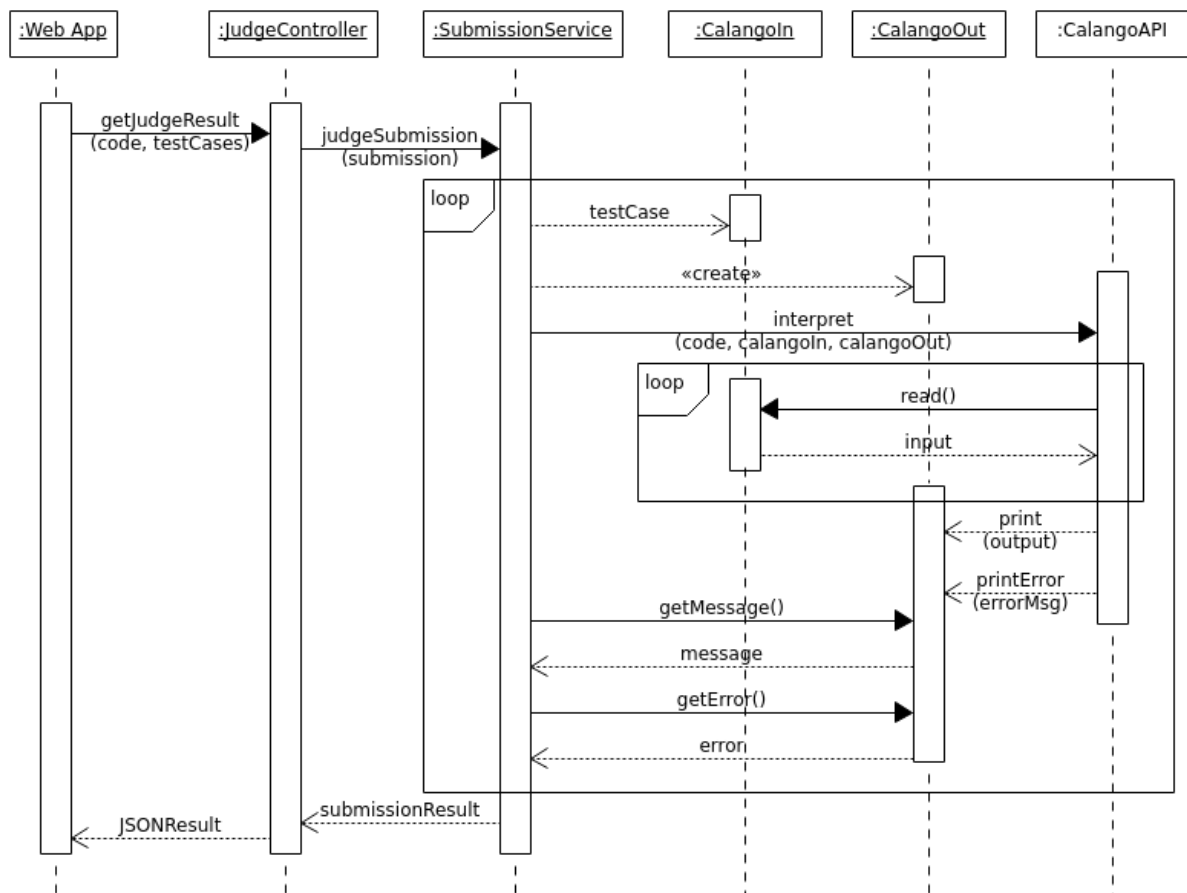
Source: the author

Figure 23 – Logical database schema generated by the ORM for PostgreSQL



Source: the author

Figure 24 – Sequence diagram showing the interaction between the judge service and the Calango Interpreter to judge a submission from the Web App



Source: the author

APPENDIX C – Data Analytics SQL

```

1
2 -- GET CLASSES IDs (14A(ID=2), CC(ID=5), DD(ID=6))
3 select * from judge_courseclass;
4
5 -- STUDENTS IN EACH CLASS
6 select identifier, count(*)
7 from judge_enrollment
8      join judge_courseclass jc on jc.id =
9      ↪ judge_enrollment.course_class_id
10 where course_class_id in (2, 5, 6)
11 group by identifier
12 order by identifier;
13
14 -- SUBMISSIONS AND STUDENTS THAT SUBMITTED
15 select jc.identifier, count(*) as subs, count(DISTINCT
16      ↪ student_id) as submitted
17 from judge_submission
18      join judge_listschedule jl on jl.id =
19      ↪ judge_submission.list_schedule_id
20      join judge_courseclass jc on jl.course_class_id = jc
21      ↪ .id
22 where jc.id in (2, 5, 6)
23 group by jc.identifier
24 order by identifier;
25
26 -- STUDENTS BY LIST CONCLUSION (100% = 25)
27 select identifier, count(student_id)
28 from (select jc.identifier, count(*), student_id
29      from judge_submission
30      join judge_listschedule jl on jl.id =
31      ↪ judge_submission.list_schedule_id
32      join judge_courseclass jc on jl.
33      ↪ course_class_id = jc.id
34 where jc.id in (2, 5, 6)
35 and result = 'ACCEPTED'

```

```
30         group by jc.identifier, student_id
31         having count(*) = 25
32         order by identifier) as icsi
33 group by identifier
34 order by identifier;
35
36 -- TRIED NON-EVALUATIVE
37 select identifier, count(DISTINCT student_id)
38 from judge_submission
39         join judge_courseclass jc on jc.id =
40         ↪ judge_submission.course_class_id
41 where jc.id in (2, 5, 6)
42 group by identifier
43 order by identifier;
44
45 -- CONCLUDED NON-EVALUATIVE
46 select identifier, count(DISTINCT student_id)
47 from judge_submission
48         join judge_courseclass jc on jc.id =
49         ↪ judge_submission.course_class_id
50 where jc.id in (2, 5, 6)
51 and result = 'ACCEPTED'
52 group by identifier
53 order by identifier;
54
55 -- SUBMISSIONS AND STUDENTS PER LIST
56 select identifier, list_schedule_id, count(*), count(DISTINCT
57         ↪ student_id)
58 from judge_submission
59         join judge_listschedule jl on jl.id =
60         ↪ judge_submission.list_schedule_id
61         join judge_courseclass jc on jl.course_class_id = jc
62         ↪ .id
63 where jc.id in (2, 5, 6)
64 group by identifier, list_schedule_id
65 order by identifier;
66
67 -- RESULTS PER LIST
```

```

64 select identifier, list_schedule_id, result, count(*)
65 from judge_submission
66     join judge_listschedule jl on jl.id =
        ↳ judge_submission.list_schedule_id
67     join judge_courseclass jc on jl.course_class_id = jc
        ↳ .id
68 where jc.id in (2, 5, 6)
69 group by identifier, list_schedule_id, result
70 order by result, identifier, list_schedule_id;
71
72 -- SUBMISSIONS BY DAY, MONTH AND STUDENT CONCLUSION (100% =
    ↳ 25)
73 select identifier,
74     list_schedule_id,
75     date_part('day', submitted_at::date) as day,
76     date_part('month', submitted_at::date) as month,
77     COUNT(*)
78 from judge_submission js
79     join judge_listschedule jl on jl.id = js.
        ↳ list_schedule_id
80     join judge_courseclass jc on jl.course_class_id = jc
        ↳ .id
81 where (jc.id = 2 and student_id in
82     (select student_id
83     from judge_submission
84     join judge_listschedule jl on jl.id = judge_submission.
        ↳ list_schedule_id
85     join judge_courseclass jc on jl.course_class_id = jc.id
86     where jc.id = 2 and result = 'ACCEPTED'
87     group by student_id
88     having count(*) = 25)
89 )
90 or (jc.id in (5, 6) and student_id in
91     (select student_id
92     from judge_submission
93     join judge_listschedule jl on jl.id = judge_submission.
        ↳ list_schedule_id
94     join judge_courseclass jc on jl.course_class_id = jc.id
95     where jc.id in (5, 6) and result = 'ACCEPTED'

```

```

96     group by student_id
97     having count(*) = 25)
98 )
99 group by identifier, list_schedule_id, day, month
100 order by identifier, list_schedule_id, month, day;
101
102 -- NUMBER OF VETERANS (registration_number < ...) AND
    ↪ FRESHMAN (registration_number > ...)
103 select identifier, count(DISTINCT student_id)
104 from judge_courseclass jc
105     join judge_enrollment je on jc.id = je.
    ↪ course_class_id
106     join accounts_student "as" on je.student_id = "as".
    ↪ id
107 where (((jc.id = 2 AND registration_number > 202000000)
108     OR (jc.id in (5, 6) AND registration_number > 211000000))
    ↪ )
109 group by identifier;
110
111 -- NUMBER OF WOMEN AND MEN (change list of registration
    ↪ numbers)
112 select identifier, count(DISTINCT student_id)
113 from judge_courseclass jc
114     join judge_enrollment je on jc.id = je.
    ↪ course_class_id
115     join accounts_student "as" on je.student_id = "as".
    ↪ id
116 where registration_number in ( --SUPRESSED )
117 group by identifier;
118
119 -- concluded all EPs (=25) AND FRESHMAN/VETERAN
120 select identifier, count(DISTINCT student_id)
121 from (select jc.identifier, student_id, count(*)
122     from judge_submission
123         join judge_listschedule jl on jl.id =
    ↪ judge_submission.list_schedule_id
124         join judge_courseclass jc on jl.
    ↪ course_class_id = jc.id
125         join accounts_student "as" on "as".id =

```

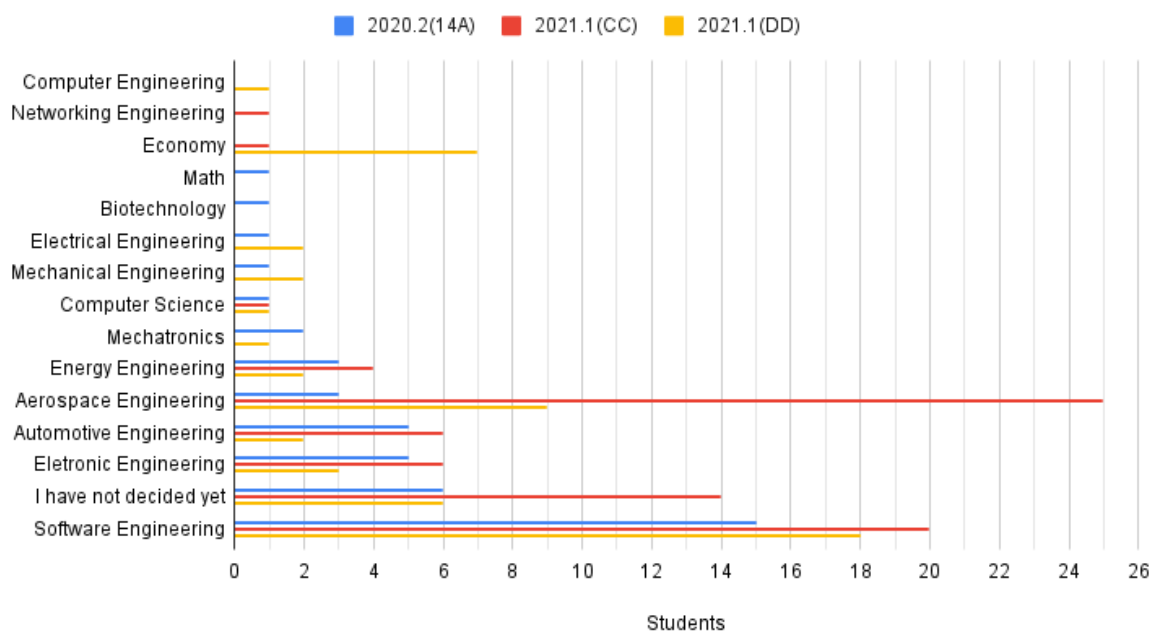
```
                                ↪ judge_submission.student_id
126     where jc.id in (2, 5, 6)
127         and ((jc.id = 2 AND registration_number < 202000000)
128             OR (jc.id in (5, 6) AND registration_number <
129                 ↪ 211000000))
129         and result = 'ACCEPTED'
130     group by jc.identifier, student_id
131     having count(*) = 25
132     order by identifier) as icsi
133 group by identifier;
```


APPENDIX D – Surveys

This chapter presents the questions of each survey and their results. It is important to remind that these questions and their options were presented to students in Portuguese.

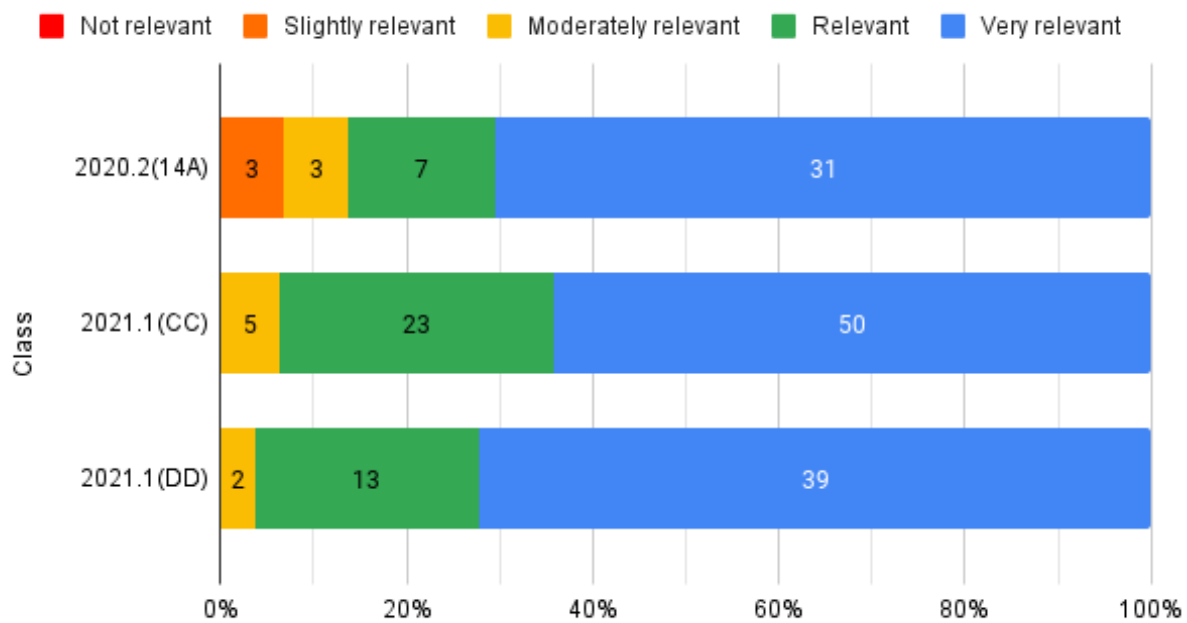
D.1 First Survey Results

Figure 25 – (S1Q1) What degree do you pursue?



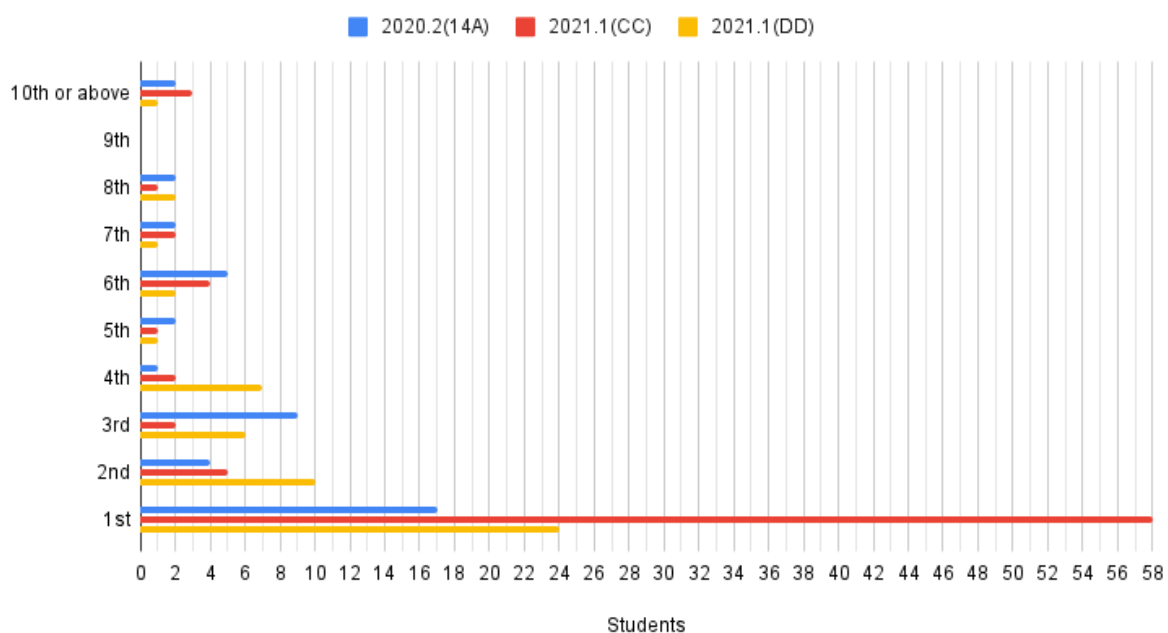
Source: the author

Figure 26 – (S1Q2) How important do you think programming knowledge is to the degree you have chosen (or intend to choose) and to your professional life?



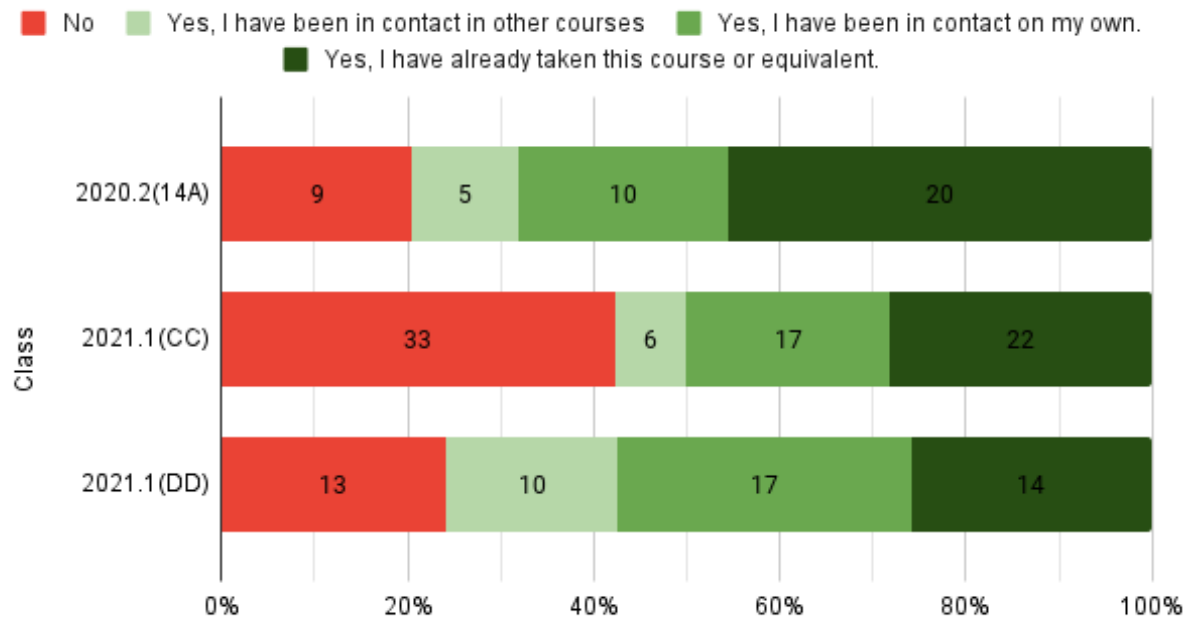
Source: the author

Figure 27 – (S1Q3) What semester of your degree are you in?



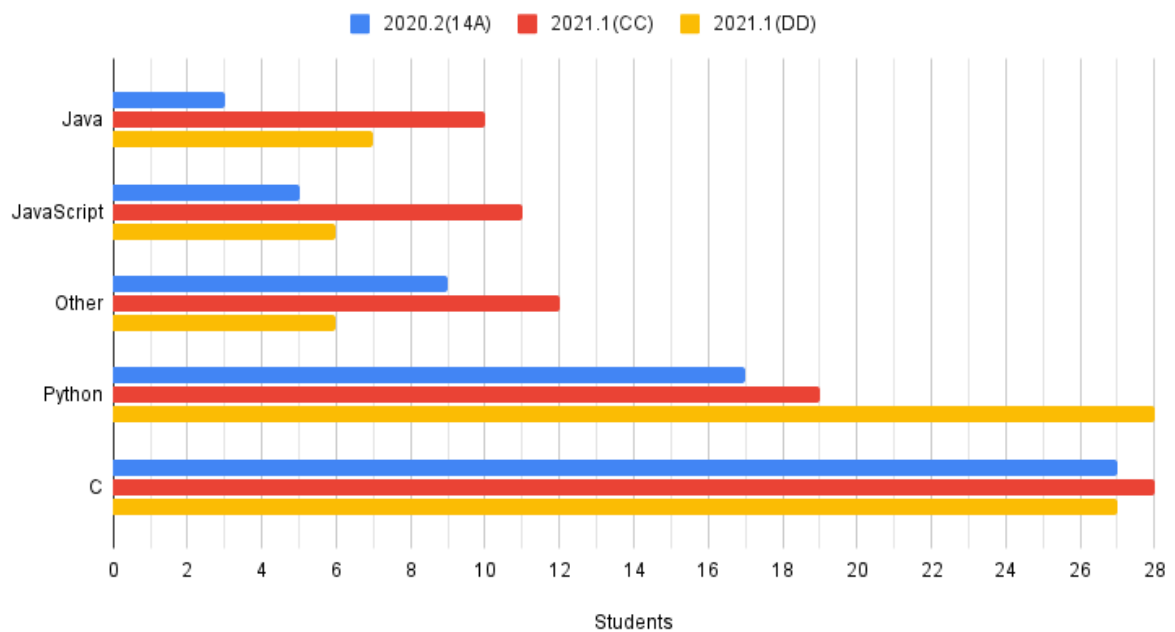
Source: the author

Figure 28 – (S1Q4) Have you ever been in contact with programming languages?



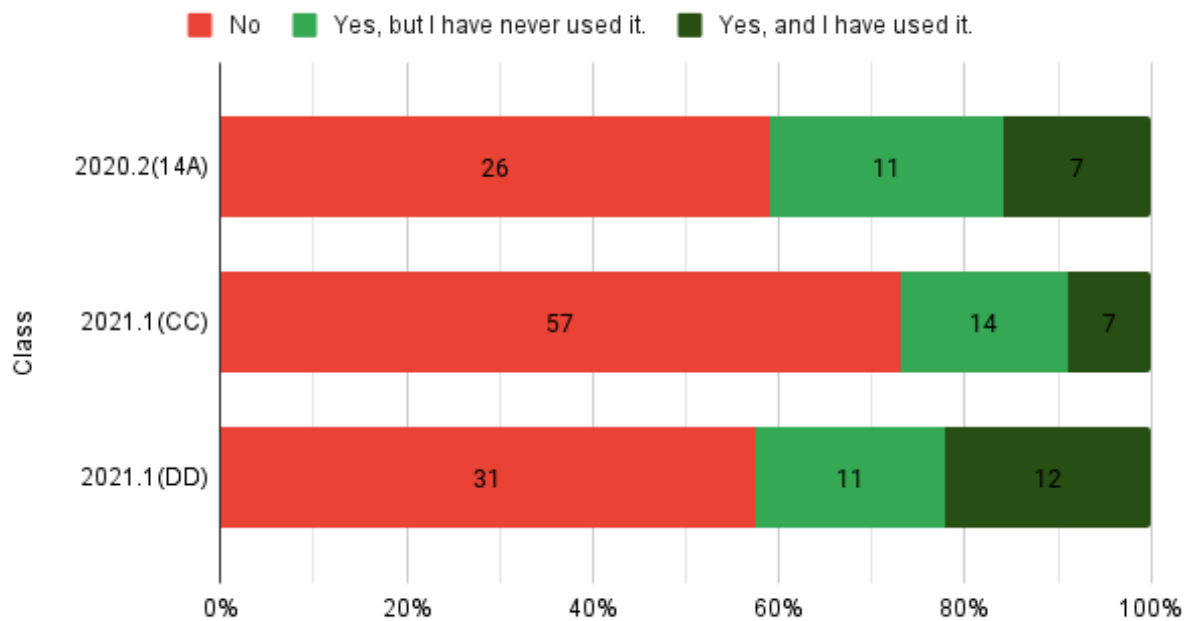
Source: the author

Figure 29 – (S1Q5) If you have been in contact with programming languages, mark the languages with which you had contact.



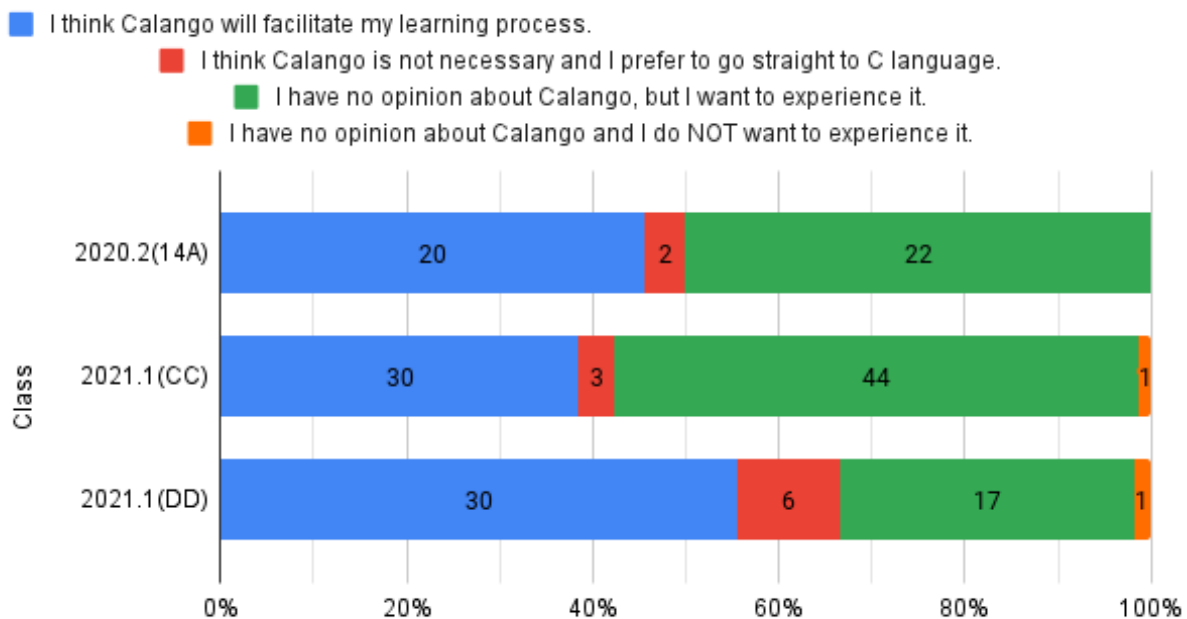
Source: the author

Figure 30 – (S1Q6) Have you ever heard of the Calango language?



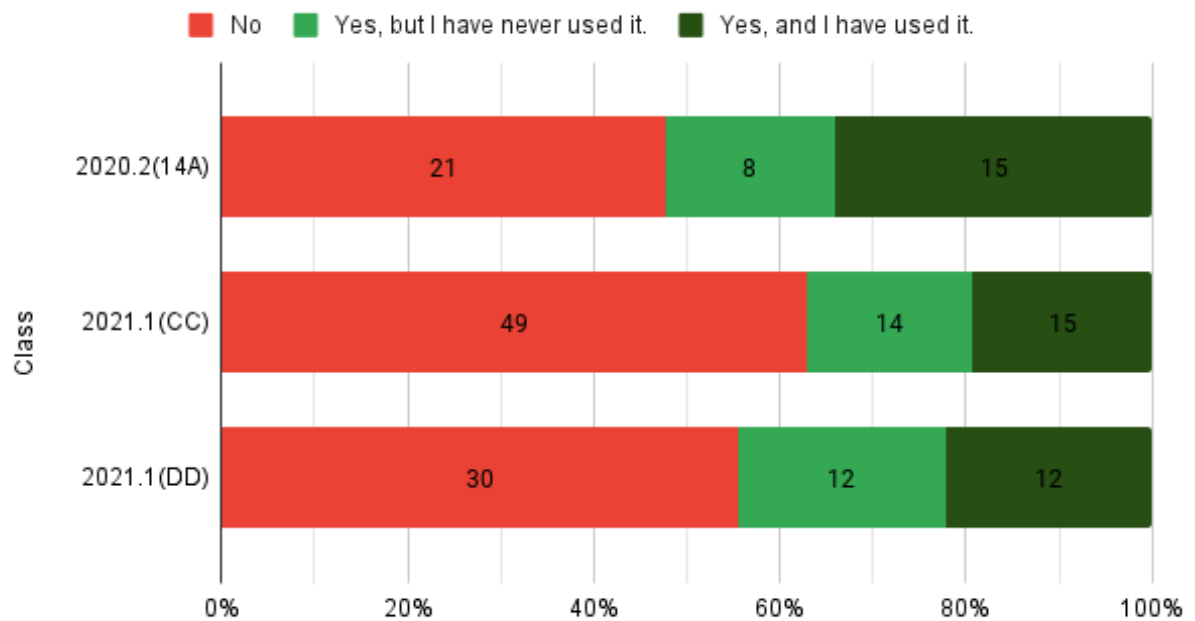
Source: the author

Figure 31 – (S1Q7) In relation to Calango and your expectations regarding the course, you...



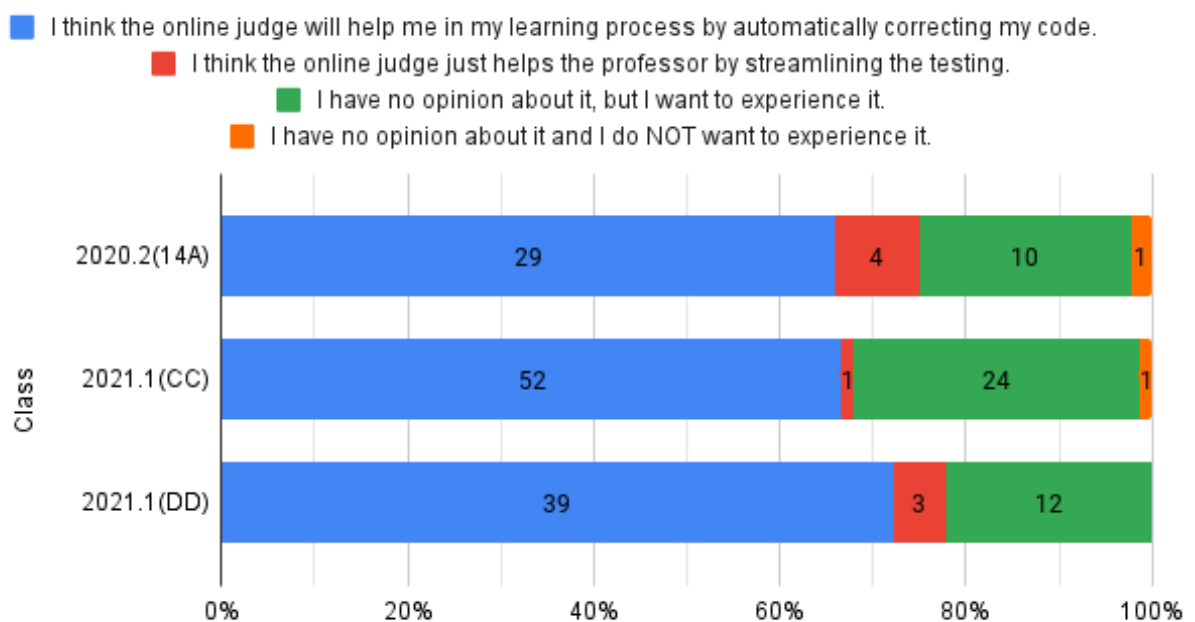
Source: the author

Figure 32 – (S1Q8) Do you know what an online judge is?



Source: the author

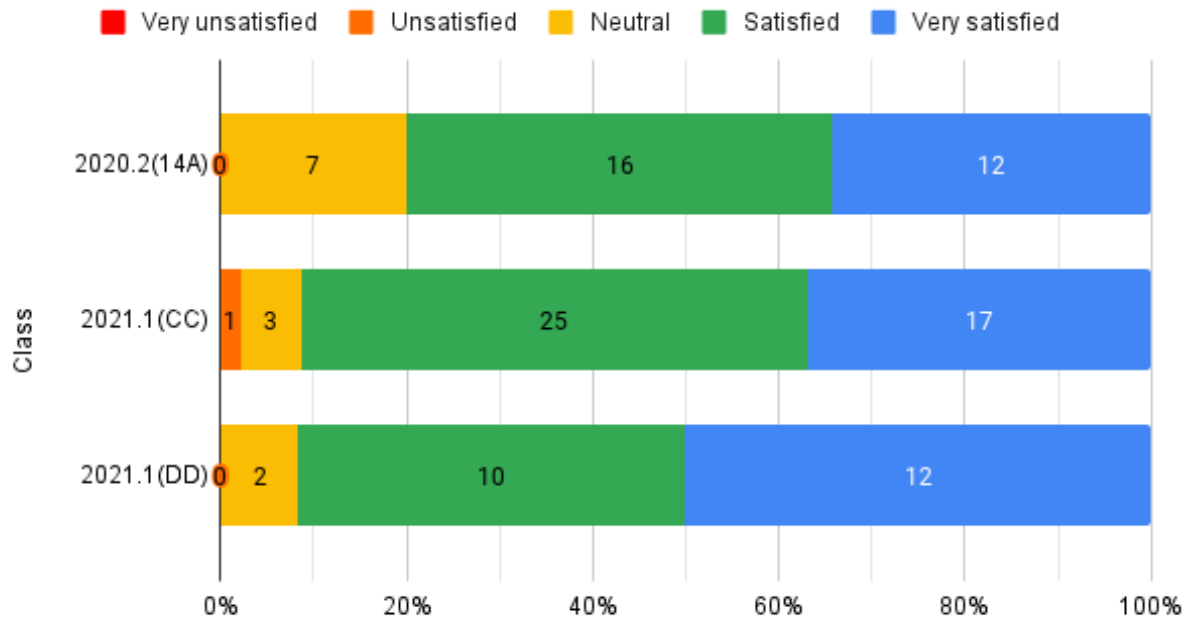
Figure 33 – (S1Q9) In relation to online judges and your expectations regarding the course, you...



Source: the author

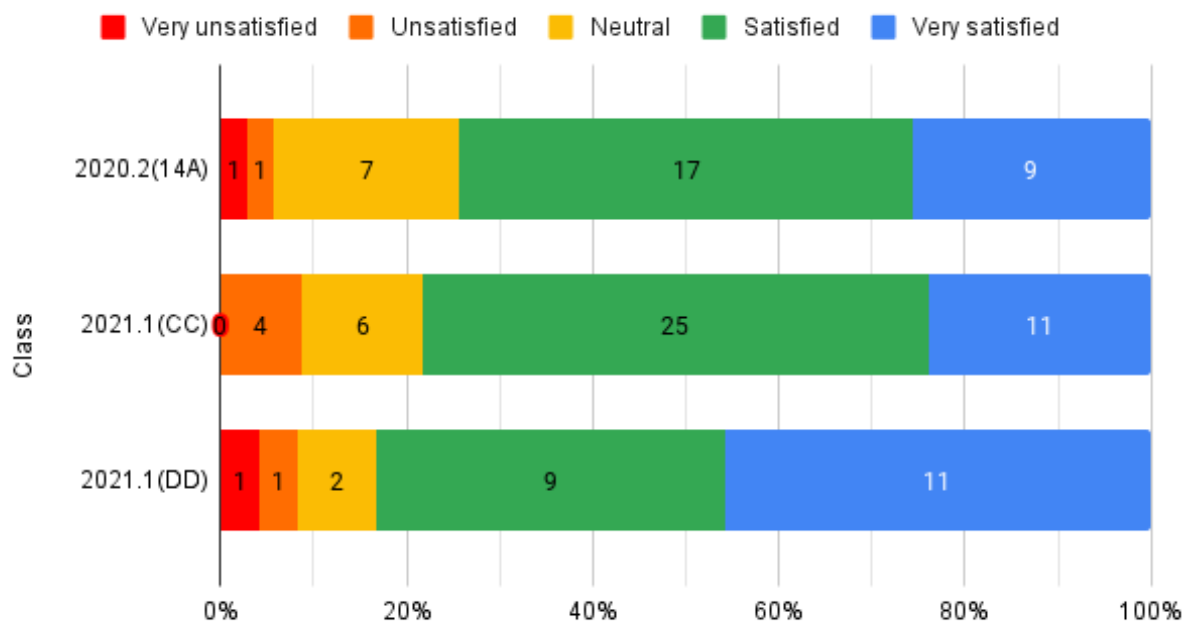
D.2 Second Survey Results

Figure 34 – (S2Q2) Regarding the use of CALANGO as a tool and programming language, in general, you were. . .



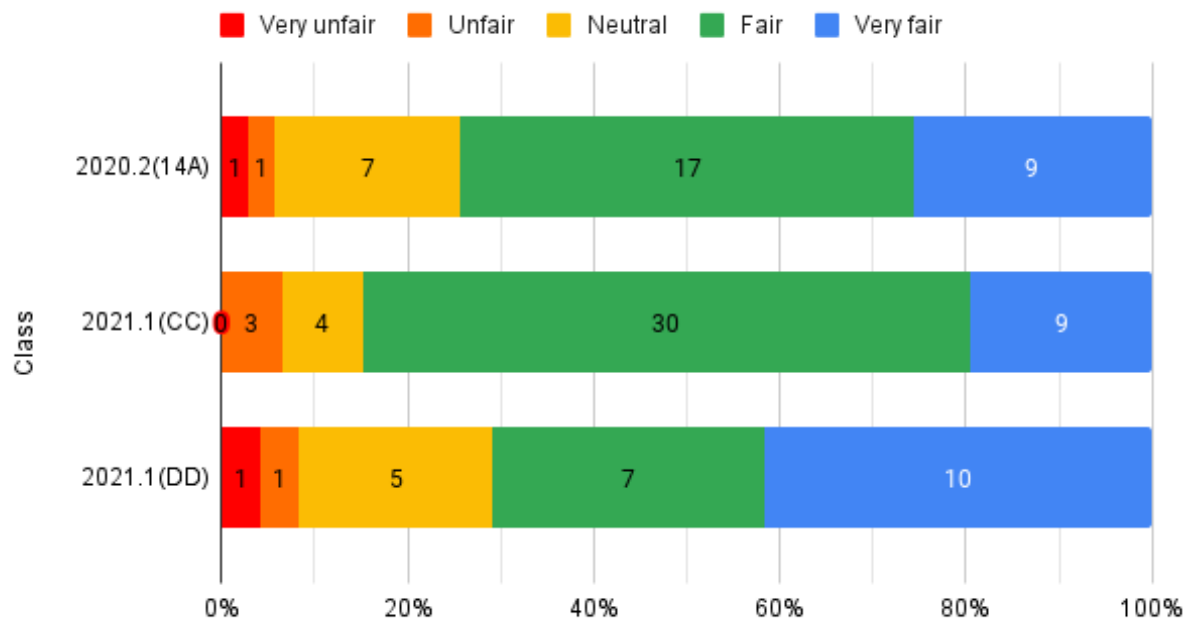
Source: the author

Figure 35 – (S2Q4) Regarding the use of COJ as an online judge, in general, you were. . .



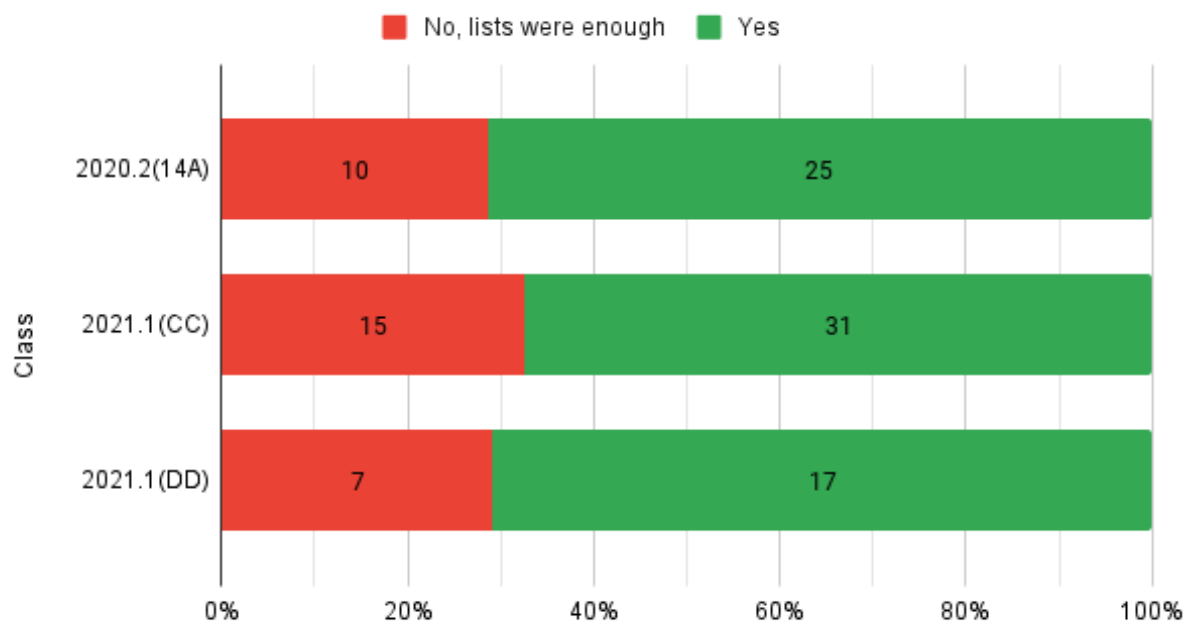
Source: the author

Figure 36 – (S2Q5) What did you think about COJ's judgment?



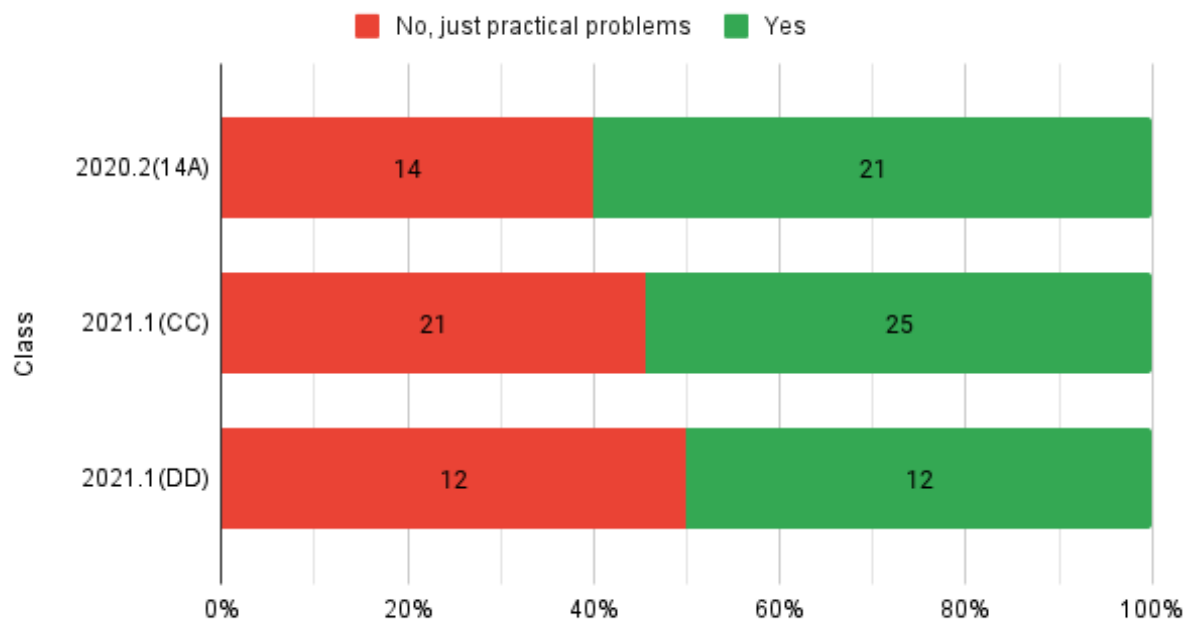
Source: the author

Figure 37 – (S2Q6) In your opinion, would it be interesting to have more questions in COJ, in addition to the lists, for practicing?



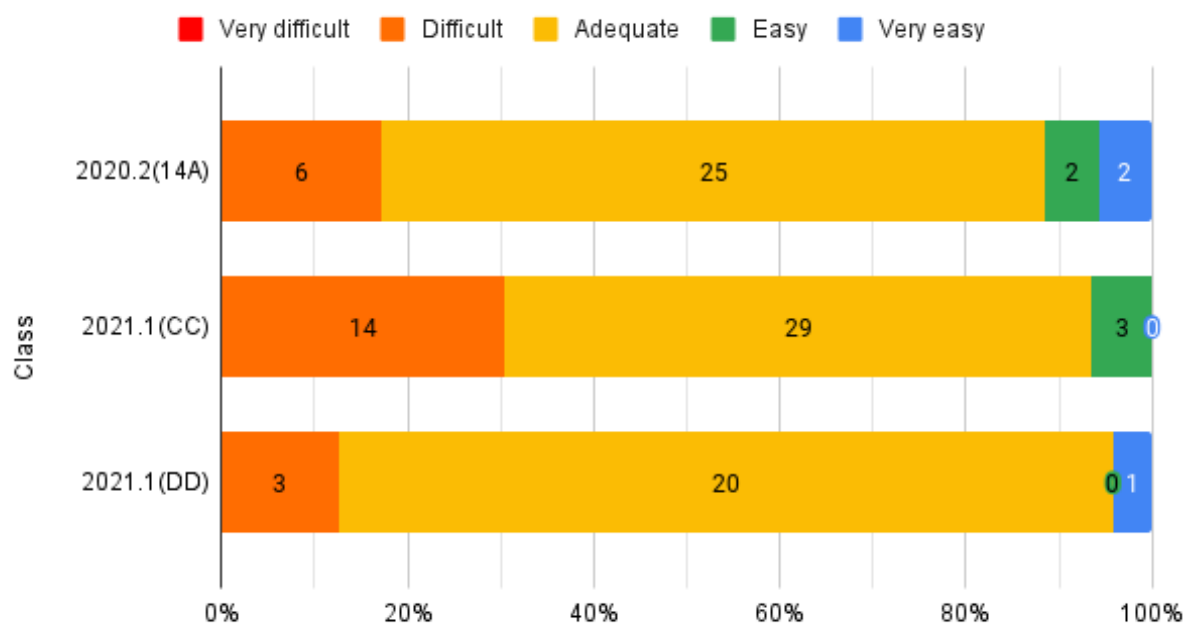
Source: the author

Figure 38 – (S2Q7) In your opinion, would it be interesting for the COJ to approach theoretical content, such as multiple-choice questions?



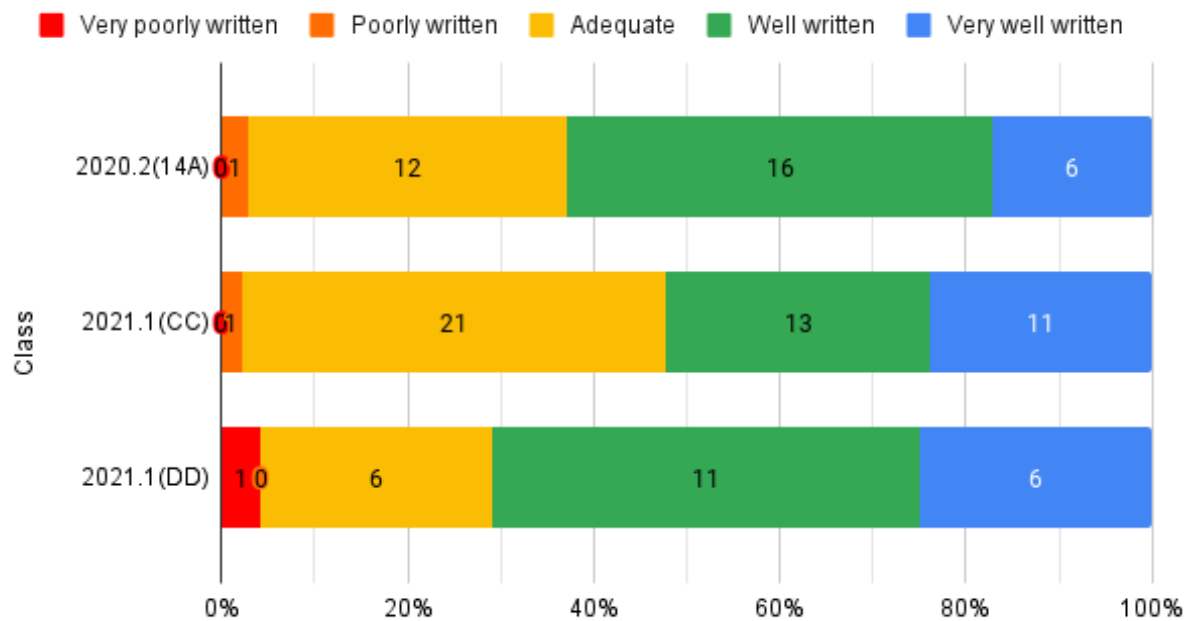
Source: the author

Figure 39 – (S2Q8) What did you think about the level of COJ problems?



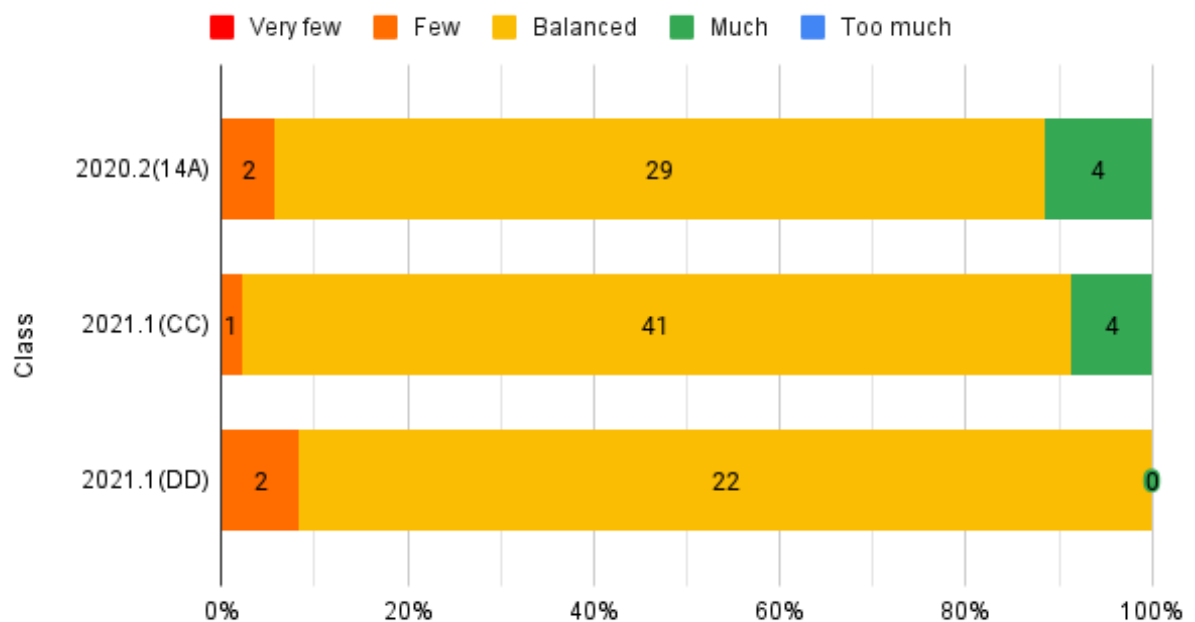
Source: the author

Figure 40 – (S2Q9) What did you think about the quality of COJ problems and their test cases?



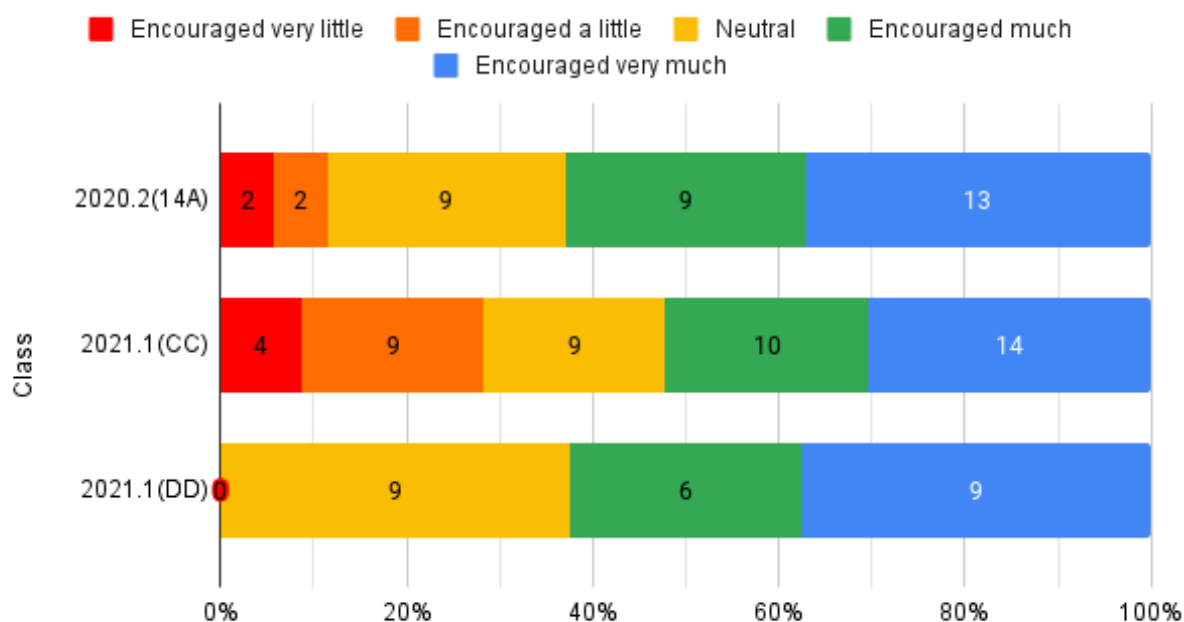
Source: the author

Figure 41 – (S2Q10) What did you think about the number of questions on the lists?



Source: the author

Figure 42 – (S2Q11) Did the submission chart on COJ's home page encourage you to make better submissions to improve your results?



Source: the author

Table 14 – (S2Q12)(2021.1(DD)) If you want to add something, use the field below to commend or suggest improvements for Calango or COJ.

Open responses
<i>"in only one proposed exercise I received wrong answer several times, in which I did 34 tests and all correct, and even with help from colleagues and monitors I analyzed my program, and without being able to identify the error I gave up submitting without understanding where I was wrong. perhaps there could be a clearer message in coj identifying the error more specifically to aid learning and correction."</i>
<i>"Improvement in the examples in the help area of calango, especially in the part of conditionals and repetitions"</i>

Source: the author

Table 12 – (S2Q12)(2020.2-14A) If you want to add something, use the field below to commend or suggest improvements for Calango or COJ.

Open responses
<i>"Certain codes output correct results in all ways but were not accepted by COJ, I believe that a system that accepts more different codes but with the same purpose, would be a better system"</i>
<i>"I believe that the COJ needs to improve the test cases because there was a case in which the problem was incorrect and was submitted."</i>
<i>"Honestly, I don't like an online judge because of the problems I have had with them the other times I have taken the course. I believe that, in a way, it also limits the students' thinking regarding training outputs (some like to ornament the outputs and for that they are "printing" several things before the final result, and I believe that this helps the student to improve the code whenever possible, but the online judge prevents it)."</i>
<i>"It would be ideal if the help box from Calango was complemented with some information, such as the possibility of choosing the number of decimal places of a number, which is not present there and makes the use of Calango more complicated than necessary."</i>
<i>"I believe that a good addition to COJ would be a representative template, providing a basis for how to make the code. It could be accessed after the lists' deadlines. This could help the student to understand some exercises that are confusing."</i>
<i>"Add extra problems (or an extra list) that are more complex (not required to be delivered), to serve as a practice for people who want to delve deeper into the programming logic"</i>
<i>"I believe that what is missing in COJ is a way to better specify the problem that is preventing the algorithm from working, pointing out precisely the problem, would help me understand the problem and be able to solve it more efficiently."</i>
<i>"I think it would be important to have a tab on the help page dedicated to the number of decimal places and their formatting in Calango."</i>
<i>"For students who arrived later, it should open more time to ask old questions, because not everyone was able to learn to program easily."</i>
<i>"I didn't like COJ very much because it had some errors and I couldn't identify the error by myself. As I already have a knowledge of programming, I found it boring to use Calango, but I believe that for beginners it must have been very useful."</i>

Source: the author

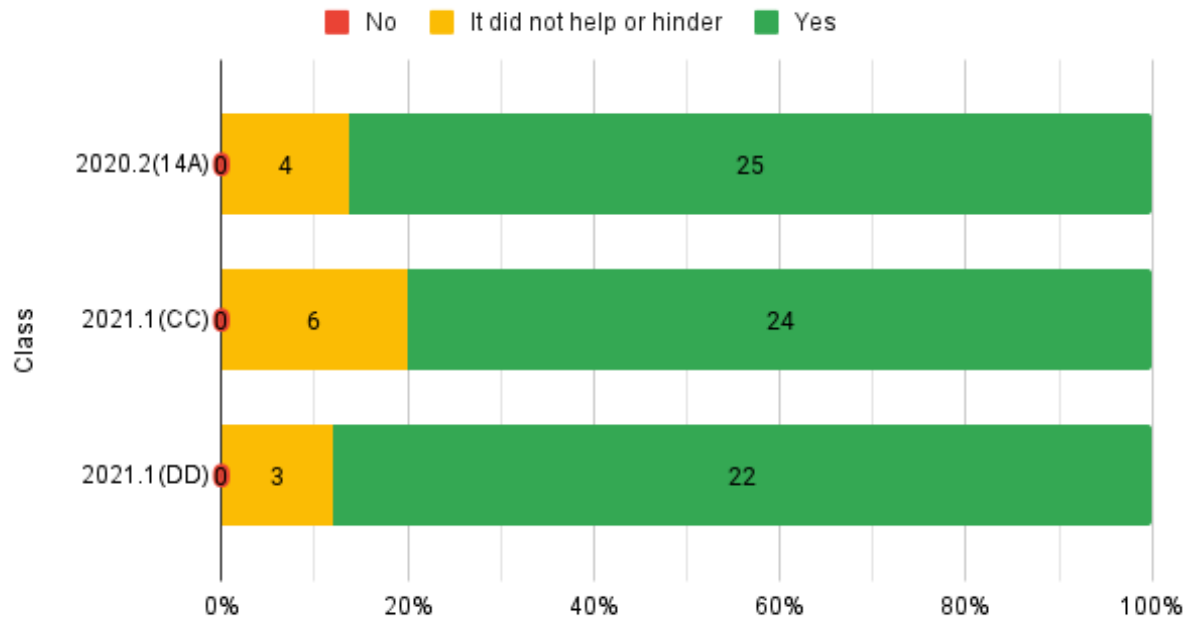
Table 13 – (S2Q12)(2021.1-CC) If you want to add something, use the field below to commend or suggest improvements for Calango or COJ.

Open responses
<i>"As a suggestion, I think it would be in everyone's interest to have an improved Calango help tab. Explain more about each function and, mainly, insert examples of how a given function is used. In this respect, I found the help tab too shallow and sometimes had to resort to other resources, which would not be necessary with a more complete "manual". Overall, Calango is a very practical platform and, although limited compared to the horizons of other programming languages, I think it's complete enough for those just starting out."</i>
<i>"COJ could consider the percentage of the code delivered, presentation error could have a grade of 80%, for example, due to the lack of a space or something like that, I understand the binary issue, however, the lists were adequate and with time to learn and do."</i>
<i>"I thought Calango was very good, especially for those who are starting to program. As for COJ, both the fact that you can't put "escreevals" to show a message that asks the user what to write and the fact that the outputs are extremely restricted gets in the way a bit, but nothing that can't be ignored. Overall, I was very pleased with this environment."</i>
<i>"It could have the option of a log to check the input values that the COJ put in our code, because in some cases, the output/input examples worked correctly but when judging, it gave an error and we didn't know what values were inputting to give error in the code for us to analyze."</i>
<i>"I think, as a future improvement, it should put what were the main errors in programming to facilitate the correction."</i>
<i>"I found, depending on the COJ list, in general, too many questions. I think if from 5 to 4 or 3 questions it would be the perfect number. In these last 2 lists I really took a long time to complete, because every question was relatively difficult and there were 5 more."</i>
<i>"Calango is very good as it helped me start my thinking about programming, as I had never had access to this content before. Regarding COJ, it is a very good judge, but I think it would be easier to understand if it showed where the error is."</i>
<i>"COJ should have optional features to show where the programmer's error is, since calango is a pseudo-language geared towards beginners, so this feature would be very didactic"</i>
<i>"It would be interesting for COJ to show the values (or value) in which an algorithm went wrong, so it would be easier to solve the problem without the need to resort to a monitor or the professor, which is a little time-consuming and ends up discouraging the resolution of the non-evaluative questions."</i>
<i>"Very good platform that made it easier to understand programming."</i>
<i>"I enjoyed learning to program first in calango and then learn in C, I thought it made learning a lot easier."</i>
<i>"I found Calango and COJ very good tools for starting programming languages to get used to the structure of coding and the requirements of detail and perfection in the construction of the algorithm. Without a doubt, it facilitated my entry into the C language, because the clash between Portuguese and C was considerably eased through Calango."</i>

Source: the author

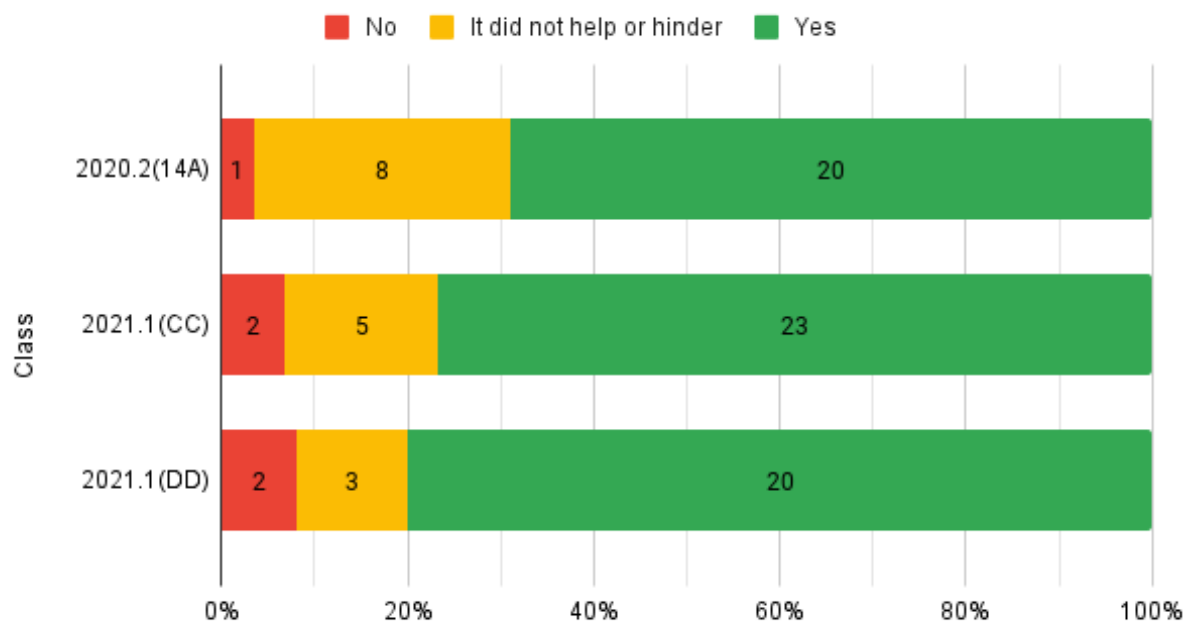
D.3 Third Survey Results

Figure 43 – (S3Q1) Did you think Calango made it easier for you to learn programming logic?



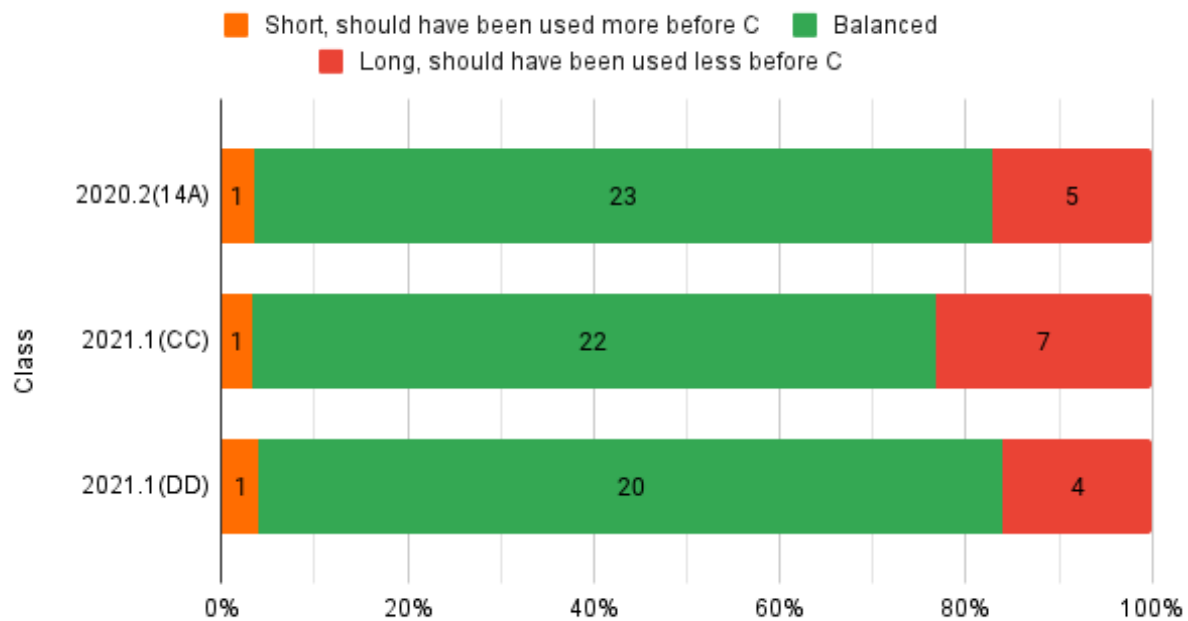
Source: the author

Figure 44 – (S3Q2) Did you think Calango made the transition to the C language easier?



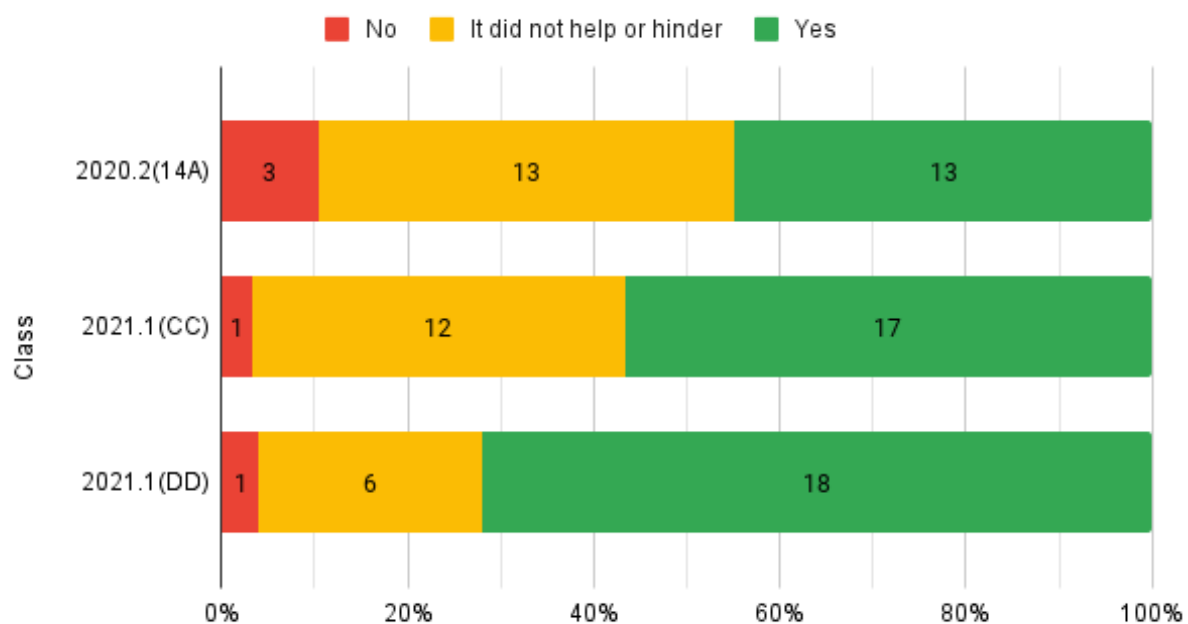
Source: the author

Figure 45 – (S3Q3) What did you think about Calango's usage time?



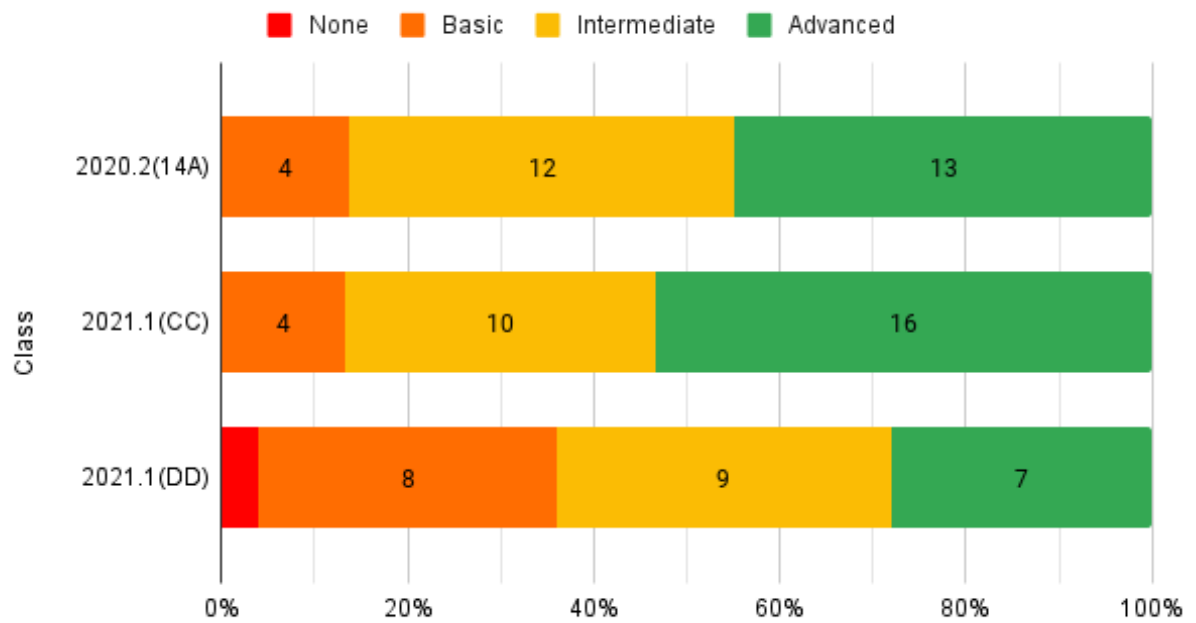
Source: the author

Figure 46 – (S3Q4) For you, did the fact that Calango is in Portuguese facilitate your learning?



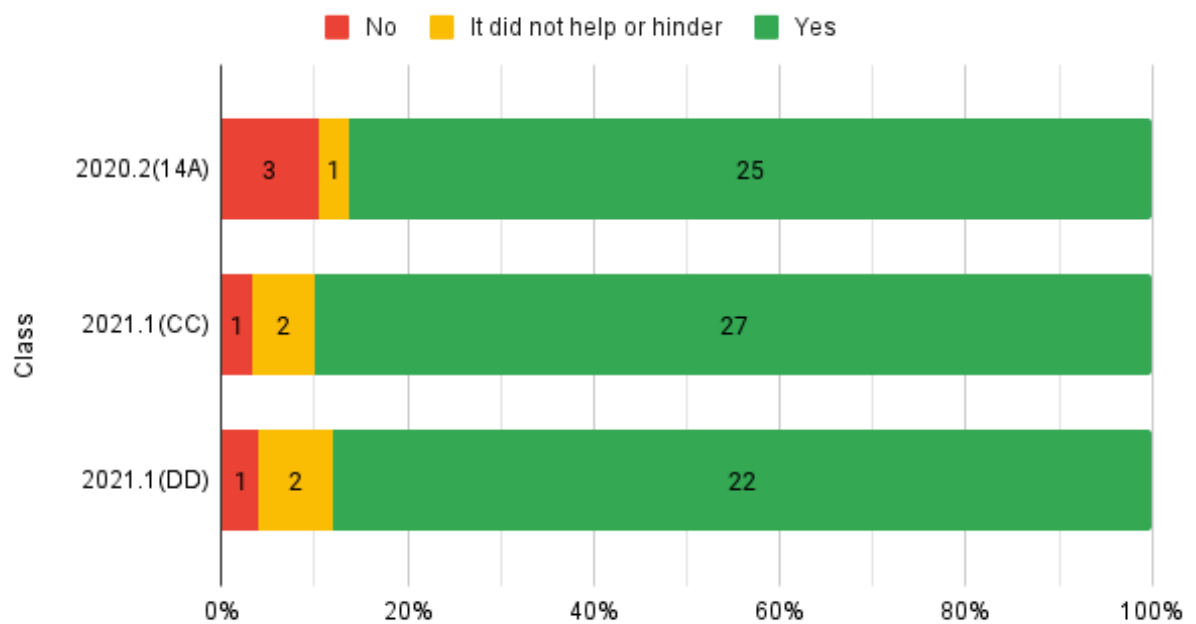
Source: the author

Figure 47 – (S3Q5) What is your level of knowledge in English?



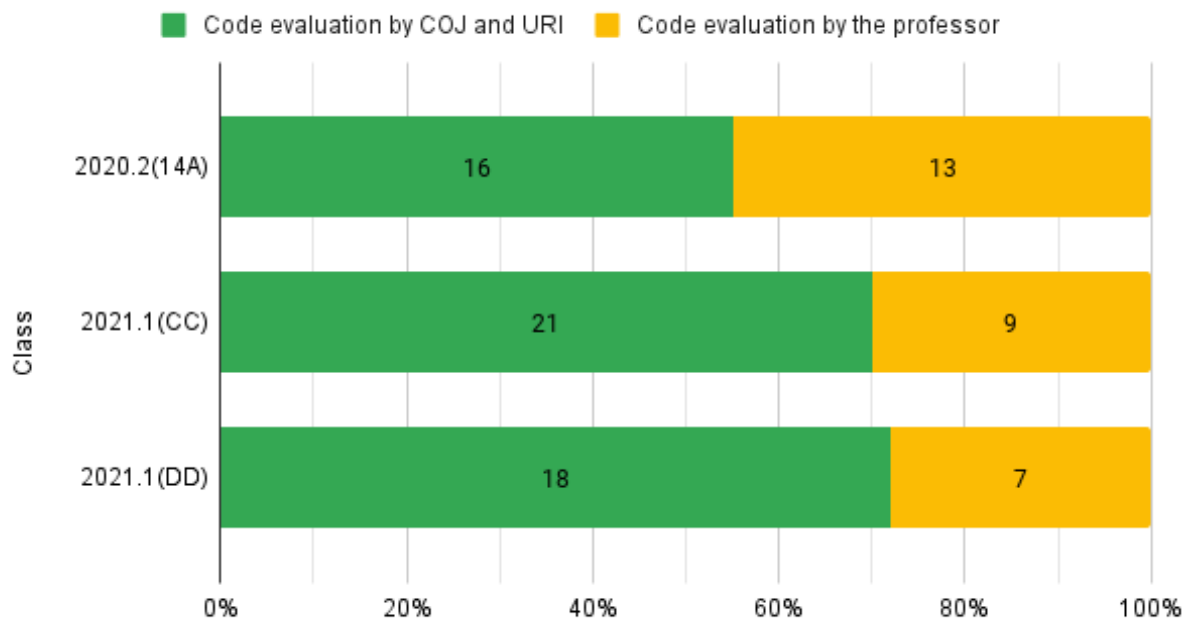
Source: the author

Figure 48 – (S2Q6) Did you find that the online judges used strengthened your practical knowledge in programming?



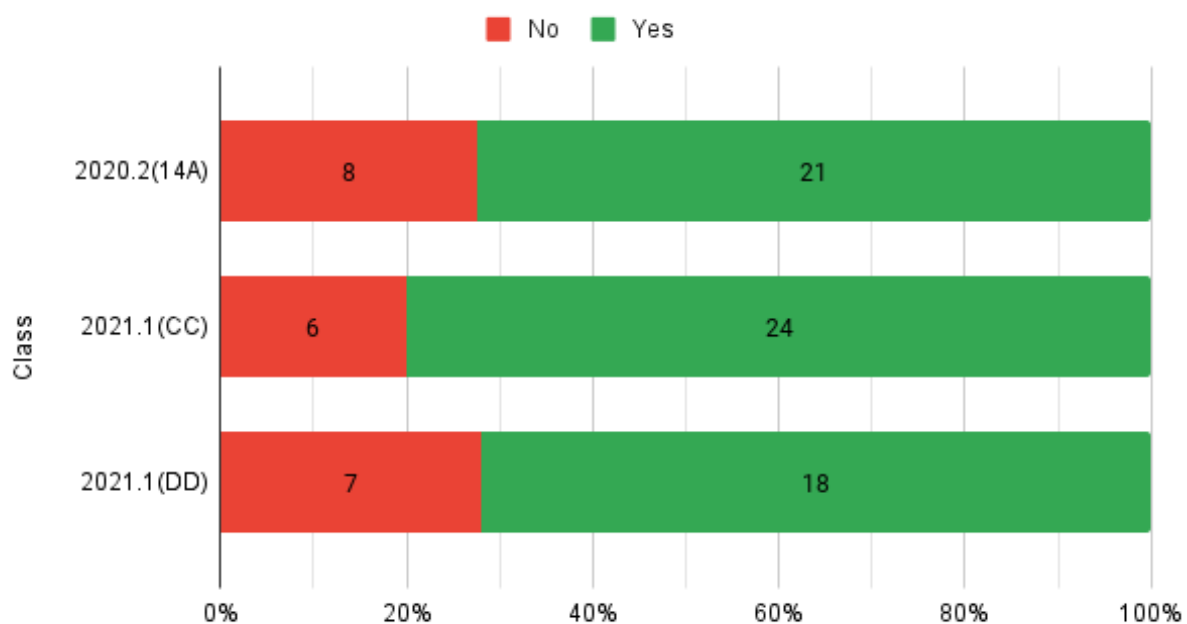
Source: the author

Figure 49 – (S3Q7) For you, which methodology makes you more motivated in the course?



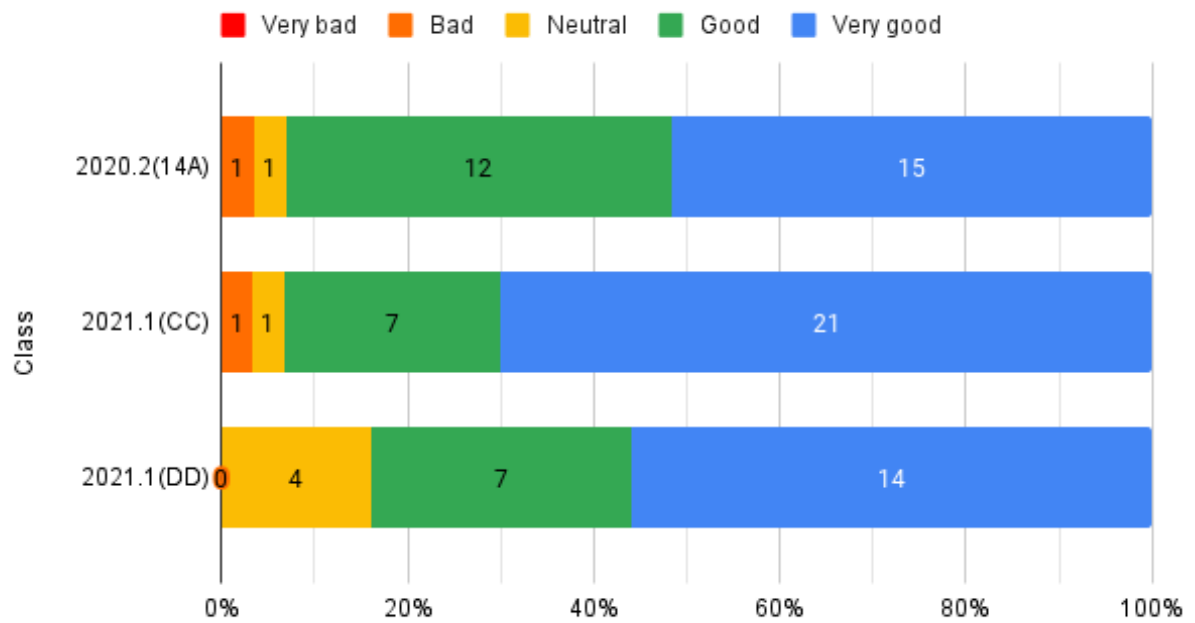
Source: the author

Figure 50 – (S3Q8) Do you think online judges (COJ and URI) have made the quality of your codes better than if you had to deliver directly to the teacher without prior feedback?



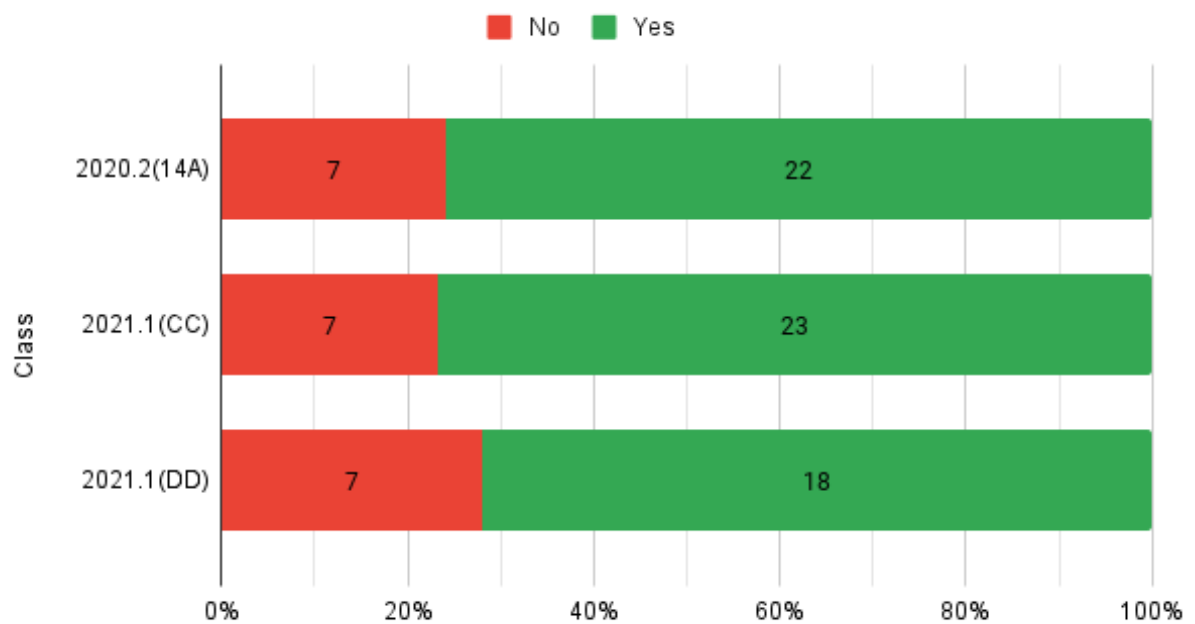
Source: the author

Figure 51 – (S3Q9) What did you think about taking the course in this class?



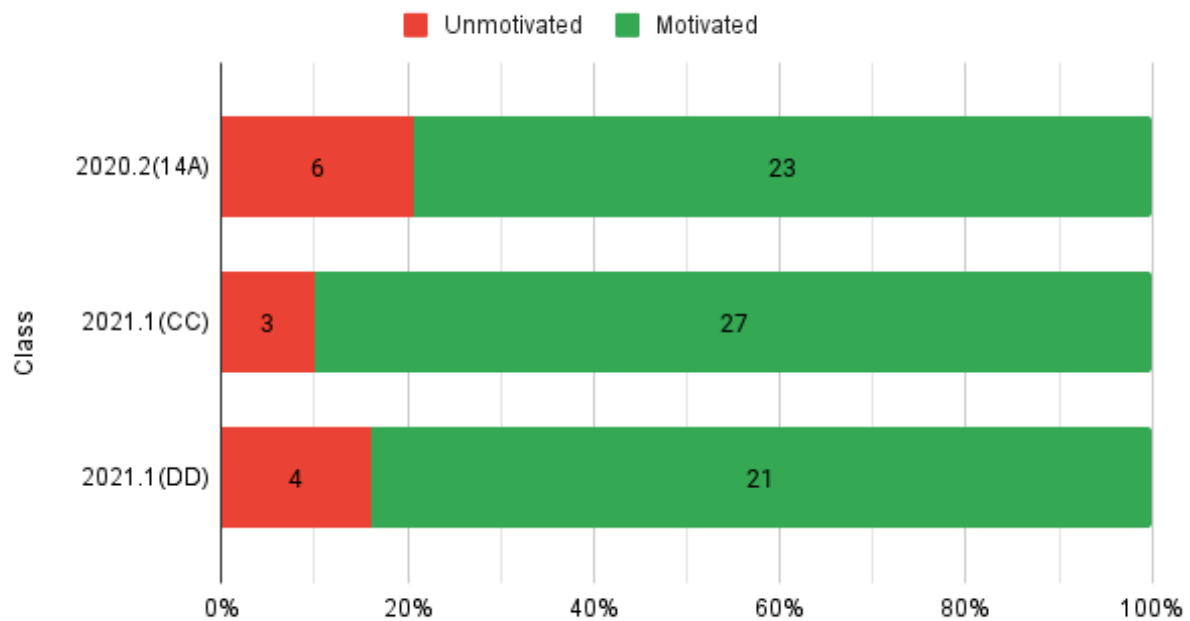
Source: the author

Figure 52 – (S3Q10) In your opinion, were all the contents of the course satisfactorily addressed?



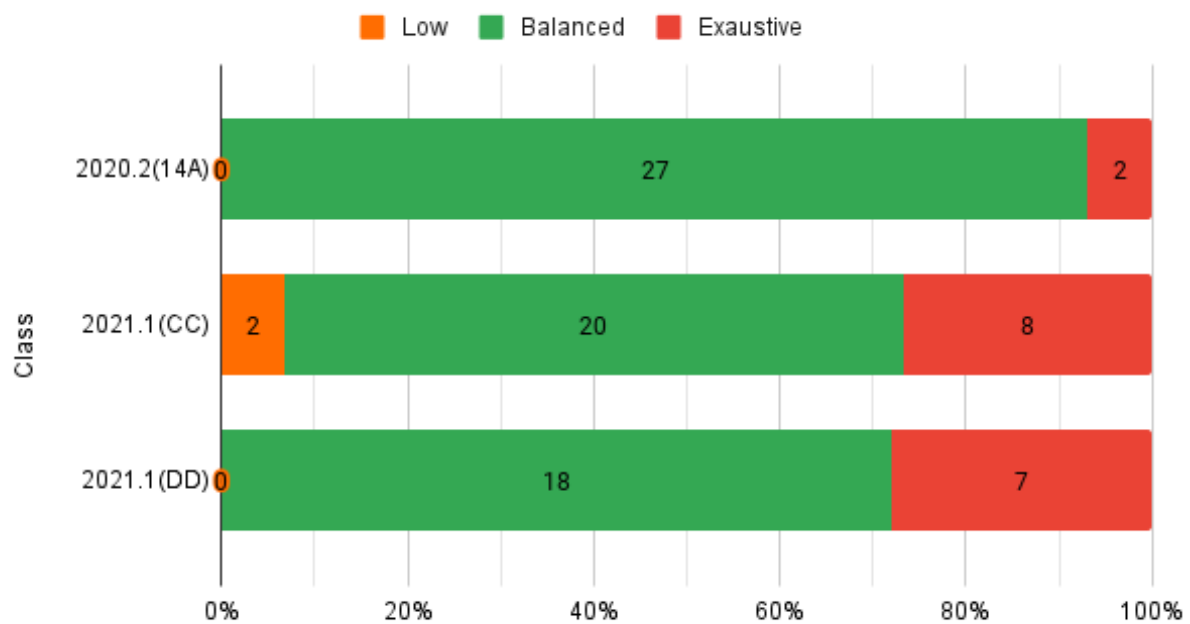
Source: the author

Figure 53 – (S3Q11) Most of the time in this course, did you feel motivated or unmotivated?



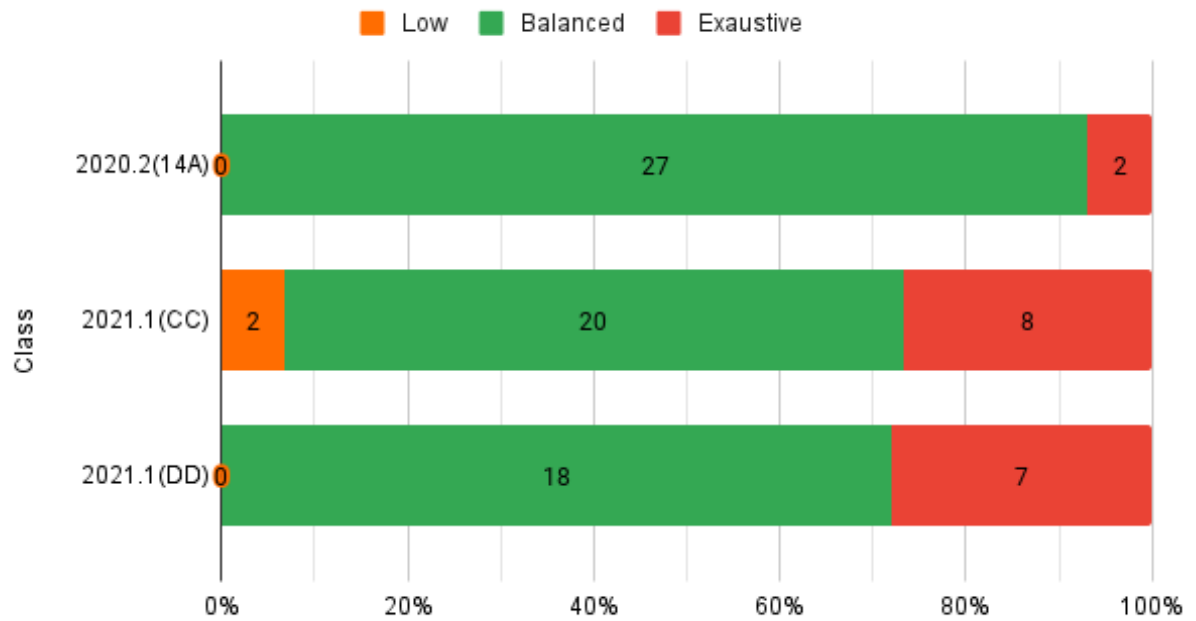
Source: the author

Figure 54 – (S3Q12) What did you think about the pace of the course?



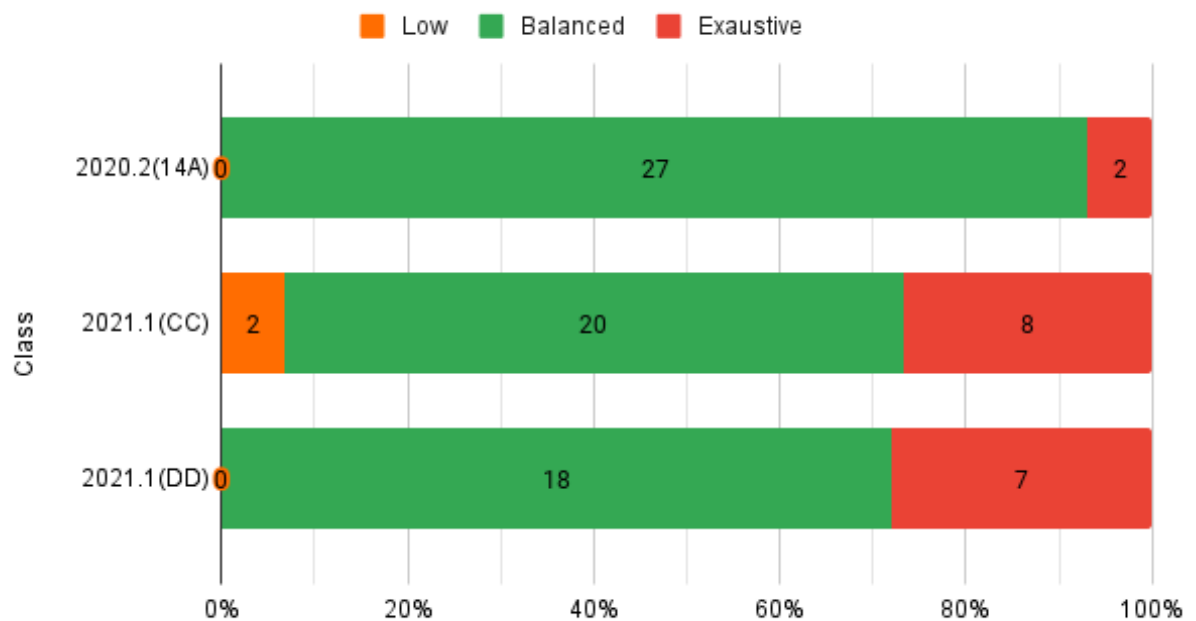
Source: the author

Figure 55 – (S3Q13)(2020.2-14A) What did you think about the workload of the course?



Source: the author

Figure 56 – (S3Q13)(2021.1-CC) What did you think about the workload of the course?



Source: the author

Table 15 – (S3Q14)(2020.2-14A) If you want to add something, use the field below to commend or suggest improvements to the course.

Open responses
<i>"This was the second time I took the course. The method that the professor approached this semester really made me learn! And the teacher was very accessible, answered the class's doubts and was understanding. I had difficulty doing the final work, but the methodology that the professor approached during the semester made me better understand what I was doing and how my code should be. And an added compliment to the monitors who did a great job too! That was the best APC class."</i>
<i>"I think remote classes have been really hard for me and for everyone, especially balancing the obligations of my personal life with college, but I believe I managed to get the support I needed to take the subject and learn a lot."</i>
<i>"The subject itself was great, but I found the content very focused on language structures and less on the logical part, and all the exercises were about logic."</i>
<i>"I think, in general, the course was very well conducted! Although I personally preferred that the C language had been presented right away, without the calango part, I recognize that it was very important, since in this part the students did not have the option of going to Google for a solution when doing exercises. It was important to train the basics without internet interference. For the rest, I wish the file manipulation part had been worked on more before the project was done, but I also recognize that researching things on my own was positive. Thanks!"</i>
<i>"I believe it was a good course and a good semester, I realized that a lot of things were left for us to search by ourselves, and I don't know if that's good or bad, but it taught me that I should go after things and I liked that. In addition, the course is good and the monitors are also very good, I did very well anyway (I don't know if I mentioned it, but at least I feel that I learned a lot)."</i>
<i>"I thought the whole experience was very good, I already knew the course syllabus in C but I would really like to have learned this method for the first time. You rocked, thanks for the semester."</i>
<i>"The APC course provided me with a good foundation to start my studies in Python and apply the knowledge acquired throughout my academic life in the field of aerospace engineering."</i>
<i>"In my opinion, the intercalation between face-to-face classes and remote classes (learning missions and classes with the teacher) was not well organized. Classroom classes were repetitive and often unnecessary. I don't remember having any significant information from the face-to-face classes that changed the way I was solving problems with what I had seen in learning. I feel that this course was completely remote, as if I had taken an online course."</i>
<i>"At times I didn't understand the algorithm or the reasoning and even going to the mentoring I couldn't understand it, which disheartened me the most. But the teacher and monitors were really great, but with the workload of other subjects, the time to study everything ended up not being enough"</i>
<i>"I believe that the file manipulation part was poorly explained, both by the pdf tutorial and by the monitor videos. I had a lot of difficulties using the basics of files, and I think I could have a mission focused on that content."</i>
<i>"Well I think I could have had more time in the C language, since it is more complete than the calango, so it would be possible to better understand its functions and how to use them"</i>
<i>"The shortening of the semester due to the atypical scenario interferes a little in the assessment of the course in relation to the pace of the content given. At certain times I thought it was a bit fast-paced because in a week, where the student needs to dedicate himself to other materials, maybe he didn't learn the content of that week in the best way, but, as I said, the semester's shortening must have influenced a lot. In addition, I would like to emphasize the importance and usefulness of the calango since APC is a first semester subject (in my opinion, this is a mistake) and many students, including me, are faced with the world of programming for the first time and now is required to have a good performance to be approved. Furthermore, I would like to immensely thank the professor for the way he conducted the course, always available to answer questions, giving all classes in a serious and extremely transparent manner in the evaluation activities."</i>
<i>"It would be interesting to add more simplified literature options and useful websites (I needed both to develop the course projects) for learning and perhaps a list of additional or challenging exercises for anyone interested in learning more on their own."</i>
<i>"Similar project to that done as a final project in C but in calango before the content transition to C would help to scale larger programs."</i>

Source: the author

Table 16 – (S3Q14)(2021.1-CC) If you want to add something, use the field below to commend or suggest improvements to the course.

Open responses
<i>"some students createad a server at Discord for freshmen who was extremely helpful. Most of the questions, especially while making the exercise lists, were about this course. Other than that, I would like to praise Professor Giovanni, I really enjoyed his presence in the WhatsApp group, always clearing up doubts. The didactics and the way the contents are passed were also perfect for the course."</i>
<i>"The course was great. I learned more about C, although I would have liked the content about procedures to have been deepened in the study materials. But apart from that, the subject is great and has a balanced workload. The missions in Moodle are important because they help students to observe their evolution."</i>
<i>"I found the leap between the weekly missions in C and the final work very big. The codes required by URI are much more for fun and training than professional application, like working in C. I understand the importance of professionalism after finishing the course, however, I think the requirement of working in C is at the level only of students who opt for software engineering, being far from the expectation of the other four engineering courses. In general I am satisfied with the course and without a doubt I will take the lessons learned here for life."</i>
<i>"Great !! I just think calango could be taught faster in the beginning to have more time with the C"</i>
<i>"I was very satisfied with the results of the course. A very patient professor with excellent teaching skills, as well as being polite and easy going."</i>
<i>"I don't know if I'm running away from reality, but I think it would be good to cover the Python language as well, as it is a very simple language to understand and one of the most used ones today. The student would come away with a very good knowledge of programming knowing the basics of Python and C."</i>
<i>"It's the goal of the project, but still, the level of C's final project is way above all throughout the year, it was an excellent and challenging experience, but perhaps too complex."</i>
<i>"Complex final project"</i>
<i>"In my opinion the classes given were great, but I think it would have been better to use Calango less, since we moved to C and it was a lot of information in a short time. I also think that the project in C could have been a little more relaxed, since I spent many nights worrying about the code and taking time away from studying other subjects to be able to finish the project and even with this extreme dedication I didn't get a result as good as i expected. In general, for me, the professor's didactic is very good and I enjoyed classes a lot, but in my opinion by improving that aspect, it would make the subject perfect."</i>
<i>"I think the final project in C was a big challenge, mainly because I didn't have any kind of contact with programming before the course, but I feel that overcoming this challenge added a lot to me and generated an extremely satisfying feeling."</i>
<i>"I think an improvement would be nice, for the Calango part, as missions are synchronous. As a lot of people (myself included) don't have any programming experience when arriving at this course, Calango was quite difficult to learn all of its hacks, and there is no online content beyond the missions. I believe that with Calango, it's worth having synchronous classes and how missions are a bonus. I don't know if it would improve a lot, but it's an idea I have that I imagine would have made my learning easier."</i>
<i>"I really liked the course and it gave me another approach to programming."</i>
<i>"One of the most complicated parts of the subject was the use of files in C, but this part did not have the necessary attention to facilitate learning"</i>
<i>"I think there are many things that I will have to study for the completion of the final project, as they were not directly addressed, I already had programming logic before college, this filled me a lot for the completion of the final project, but I have reports of colleagues who did not like the teaching method, in their words, it was very "DIY", do-it-yourself."</i>
<i>"I found the course interesting, however, I think it could greatly reduce the time spent in Calango and go straight to C. That way, I would have more time to introduce and use more useful C's libraries."</i>

Source: the author

Table 17 – (S3Q14)(2021.1-DD) If you want to add something, use the field below to commend or suggest improvements to the course.

Open responses
<i>"The final project in C was extremely demanding and even though I had a good grasp of the language in C, it was too extensive work, forcing some students to give up other subjects to be able to do the work, I believe that in the coming semesters, this work could be reduced, or charged to students specifically in the software area."</i>
<i>"I'm extremely satisfied with the course, I'm going through the subject for the second time and I brought feelings of hatred with the content, I found it difficult, I didn't understand anything, and this time I not only liked the methodology, I actually learned and even fell in love with it. area, very grateful for all the willingness and commitment of the teacher and monitors."</i>
<i>"There could have been URI missions on some topics that were needed in the final project."</i>
<i>"Excellent professor, the only point that makes it difficult is the final work in C, where we have to learn some content on our own, which is frustrating at times, as finding quality material is a little complicated, besides that, excellent course."</i>
<i>"My first contact with programming, I thought it was an excellent course, I just wish some points had more time"</i>
<i>"The final project is very extensive and complicated, as there were no exercises or classes just for this before starting the project, it complicated a lot because there was a lot that the students had to learn on their own or wait for the professor to explain the class, which made it even more difficult plus the work, what would be nice is before starting the work, having classes to explain each point, what can go wrong and what usually works, the project itself added a lot, but it was a lot of work."</i>
<i>"Calango greatly facilitated my understanding of loops!!!"</i>
<i>"I think the methodology used was very efficient and turned what could have been a difficult and tiring course into an even fun one."</i>
<i>"In my opinion, the introduction to calango could have been made a little faster so it could have classes focused on files and structs."</i>
<i>"Overall, the course is very well conducted, but the final project is very demanding. I believe that the level of the project does not match the content given. Over the time we do the work, there are still things we have to learn to be able to apply it, and when the deadline comes, it's when the most important content is given to make the code work as it should (struct), and we don't have a class of all the content in which we need to apply, which makes it even more difficult (files).</i>
<i>"Professor, I found the pace of the course amazing and how the contents were approached and properly explained, but I think the struct and files part of the work was little explained and fast, it was quite complicated for me at least, and tiring. But I understand that the intention was to turn around a little. I think if I had a struct list before, in the URI, it would have helped a lot."</i>
<i>"Excellent course. In 7 semesters of UnB, it was certainly one of my favorites! Balanced and with a good amount of C elements for beginners, without overloading. The calango was excellent for learning. I would say that 80% of the learning was allocated to the project in C. Extensive and laborious, but with more than enough time and support. Congratulations to Professor Giovanni and monitors for their understanding and availability throughout the semester."</i>
<i>"I missed a bigger evaluation in the final C work, I ended up working many hours on it and I couldn't present everything I wanted about the code in the presentation. Another fact that made me upset was that I would like feedback directly from the professor about the code, although the monitor evaluated it well. Anyway, I learned a lot that I will definitely take along my learning experience!!"</i>

Source: the author

Annex

ANNEX A – SUS questionnaire

A.1 Original SUS

Figure 57 – Original SUS questionnaire

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	1	2	3	4	5
2. I found the system unnecessarily complex	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	1	2	3	4	5
3. I thought the system was easy to use	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	1	2	3	4	5
5. I found the various functions in this system were well integrated	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	1	2	3	4	5
6. I thought there was too much inconsistency in this system	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	1	2	3	4	5
8. I found the system very cumbersome to use	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	1	2	3	4	5
9. I felt very confident using the system	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	1	2	3	4	5

Source: [Brooke \(1996\)](#)

A.2 European Portuguese SUS

Table 18 – Validated European Portuguese version of the SUS questionnaire

Original Item	Corresponding item in Portuguese
I think that I would like to use this system frequently.	Acho que gostaria de utilizar este produto com frequência.
I found the system unnecessarily complex.	Considerei o produto mais complexo do que necessário.
I thought the system was easy to use.	Achei o produto fácil de utilizar.
I think that I would need the support of a technical person to be able to use this system.	Acho que necessitaria de ajuda de um técnico para conseguir utilizar este produto.
I found the various functions in this system were well integrated.	Considerei que as várias funcionalidades deste produto estavam bem integradas.
I thought there was too much inconsistency in this system.	Achei que este produto tinha muitas inconsistências.
I would imagine that most people would learn to use this system very quickly.	Suponho que a maioria das pessoas aprenderia a utilizar rapidamente este produto.
I found the system very cumbersome to use.	Considerei o produto muito complicado de utilizar.
I felt very confident using the system.	Senti-me muito confiante a utilizar este produto.
I needed to learn a lot of things before I could get going with this system.	Tive que aprender muito antes de conseguir lidar com este produto.

Source: [Martins et al. \(2015\)](#)