INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Plataforma de Software para Gestão dos Doentes Mentais Internados em Instituições do Setor Social no Hospital Fernando Fonseca

Bruno Alexandre Gomes da Silva Canelas

Licenciado

Projecto Final para obtenção do Grau de Mestre
em Engenharia Informática e de Computadores

Orientadores : Doutor Miguel Gamboa
Doutor Pedro Vieira

Júri:

Presidente: Doutor José Manuel de Campos Lages Garcia Simão

Vogais: Doutor Filipe Bastos de Freitas
Doutor Fernando Miguel Gamboa de Carvalho

Novembro, 2022

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Plataforma de Software para Gestão dos Doentes Mentais Internados em Instituições do Setor Social no Hospital Fernando Fonseca

Bruno Alexandre Gomes da Silva Canelas

Licenciado

Projecto Final para obtenção do Grau de Mestre
em Engenharia Informática e de Computadores

Orientadores : Doutor Miguel Gamboa
Doutor Pedro Vieira

Júri:

Presidente: Doutor José Manuel de Campos Lages Garcia Simão

Vogais: Doutor Filipe Bastos de Freitas
Doutor Fernando Miguel Gamboa de Carvalho

Novembro, 2022

Aos meus pais

Acknowledgments

I would like to thank both my supervisors, Fernando Miguel Gamboa de Carvalho and Pedro Vieira for the guidance they provided me while developing this project. All of your feedback and teachings have made me achieve my goals. Thank you for all the time you spend on this project with me.

I would also like to thank the members of *Hospital Fernando Fonseca* who help and guided the development of this project and provided me with the necessary tools to develop an application that would live up to their standards.

Additionally, I would like to thank all my friends who have helped me complete this journey.

Finally, I would like to thank my family who devoted so much to me and made want to achieve something better for myself and for them. They gave me everything and I hope I can live up to their expectations.

Abstract

Hospital Fernando Fonseca is an organization that manages patients with mental diseases sent to Care Houses that are able to take care of them for 24 hours a day. These patients can be sent to the Care Houses by the hospital itself or by other institutions. Once a patient has entered a Care House, the HFF is responsible for the payment of the daily cares of the patient. In case the patient comes from another institution, then it is the institution's responsibility to make these payments.

Once in the Care House, a patient might need to go to other institutions, such as hospitals, for other health related reasons. During these days, the care of the patients should not be billed since they are not in the Care House. This however, poses a problem because the HFF cannot track these visits since they are only managed by the Care House, which can lead to the days being billed without the knowledge of the HFF.

The current process of referring a patient to a Care House and managing their internment is made manually, and the communication between the HFF and the Care Houses is made via e-mail. This makes the process slow and prone to failure.

The development of this project has the goal of creating and application that will allow the different departments that take part in this process to communicate better with each other in order to simplify the process and guarantee the correct billing for the care of the patients.

This document will present the different stages of the development of the project, such as defining the requirements, creating a set of mock-ups to explain the ideas to the HFF, developing the database models, creating the application, testing it and finally deploying it.

Keywords: HFF, Care House, Invoice, Python, Django, PostgreSQL, Nginx, Gunicorn, ASGI, WSGI

Resumo

O Hospital Fernando Fonseca é uma organização que trabalha com doentes com doenças mentais os quais são enviados para Casa de Saúde de modo a serem cuidados durante 24 horas por dia. Estes pacientes podem ser enviados para as Casas de Saúde por parte do próprio HFF ou por instituições externas. Quando um paciente entra numa Casa de Saúde, o HFF ou a instituição que enviou o paciente, é reponsável pelo pagamento dos cuidados diários deste paciente.

Já dentro da Casa de Saúde, pode ser necessário que um paciente visite outras instituições, por exemplo hospitais. Durante este periodo, o HFF não pode ser cobrado pelo cuidado destes pacientes uma vez que estes não estão na Casa de Saúde. Dado que o HFF não controla estas saídas, podem ser cobrados dias nos quais o paciente não esteve na Casa de Saúde.

O processo atual de referenciação de um paciente para uma Casa de Saúde, bem como a gestão do mesmo, é realizado de forma manual, através do envio de emails entre departamentos. Este processo pode ser lento e sujeito a falhas.

O desenvolvimento deste projeto tem como objetivo a criação de uma aplicação que facilite a comunicação entre os diferentes departamentos envolvidos no processo de internamento, bem como simplificar a criação e gestão do processo e garantir que o HFF está a ser cobrado o valor real dos cuidados.

Neste documento serão apresentadas as diferentes fases de desenvolvimento do projeto tais como o levantamento de requisitos, a criação de *mock-ups* para exposição de ideias, criação de modelos de base de dados, criação da aplicação, testes realizados e por fim o *deployment*.

Palavras-chave: HFF, Casa de Saúde, Recibo, Python, Django, Nginx, Gunicorn, WSGI, ASGI

Contents

List of Figures

List of Listings

1

Introduction

PSM (in portuguese "Plataforma de Saude Mental") is developed in the context of national mental health program and aims to implement a software tool to support the reference and internment process of patients managed by the *Hospital Professor Doutor Fernando Fonseca*.

1.1 Scope

The PSM application was developed in order to be used by the HFF and its Care Houses. Currently there is an entity denominated CNSC (in portuguese "*Coordenação Nacional de Saúde Menatal*") which is in charge of 5 regional organizations, namely ARS (*Área Regional de Saúde*) Norte, ARS Centro, ARS Lisboa e Vale do Tejo, ARS Alentejo and ARS Algarve. One of the partners of this project is the coordinator of ARS Lisboa.

The application is a prototype that is planned to be used by the Lisbon regional organization in the future. Currently, the application will be tested using only the staff of the HFF and the Care Houses, however it is being designed with the intention of expanding to the Lisbon district.

1.2 Main Goals

Mental patients are just one of the types of patients that the "*Hospital Fernando Fonseca*" (HFF) is responsible for. The patients are diagnosed in the hospital, or another institution and must be taken care of.

However, the hospital does not have the means to take care of these patients. Given this fact, the hospital requires the help of external institutions denominated Care Houses (e.g. *Casa de Saúde do Telhal*) that are equipped and prepared to take these patients in and give them the necessary care and attention.

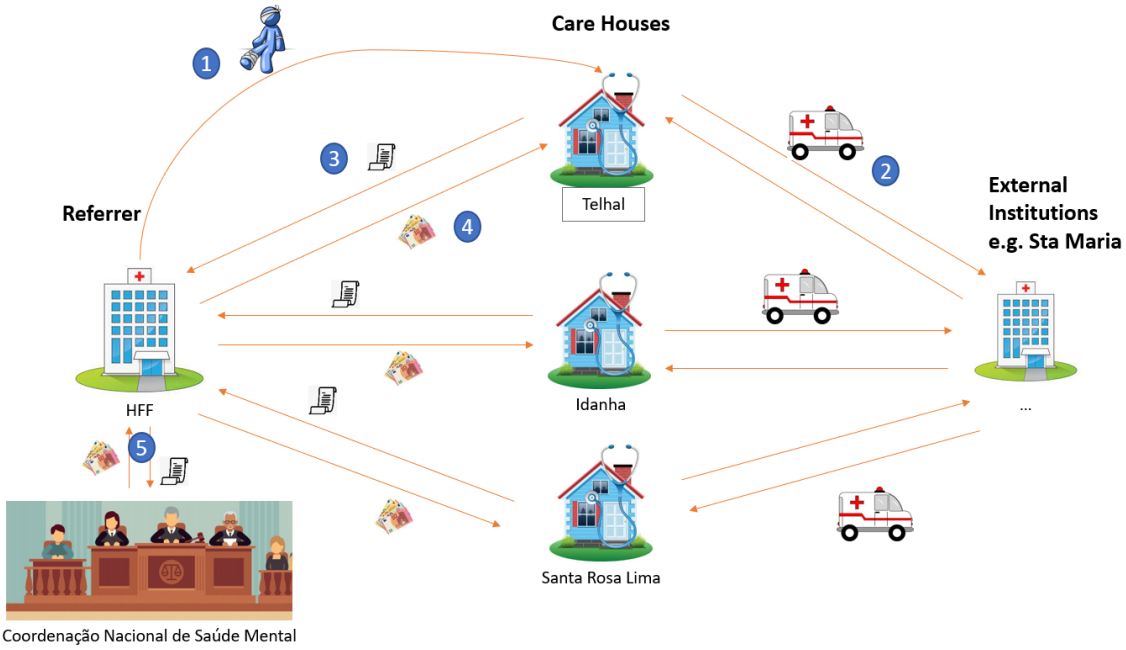

Figure 1.1: Process Description

Figure 1.1 is a high-level view of the process of a patient referral, followed by the internment process and the financial management. Once a patient has been diagnosed with a mental illness the medical staff in the HFF will need to reference the patient to a Care House. There are 3 types of typologies for referencing patients: Typology I, where the patient has been in the care house before the creation of the partnership between the care house and the HFF; Typology II where the patient is referenced by an institution other than the HFF (e.g. *Hospital de Santarém*); and finally Typology III where the patient is referenced by the HFF. If the referral is approved, the patient will be transferred to the Care House (Step 1 in Figure 1.1). Currently, the communication between the HFF and the Care Houses is done via e-mail. If an e-mail is not seen by

one of the members involved in the process, the process will stagnate, will probably bee restarted.

Once in the Care House, the patient will then be treated and accompanied by the staff in the Care House and, if necessary, taken to external institutions for other types of cares (e.g. Hospital Santa Maria) (Step 2 in Figure 1.1). The management of patients in the Care House is made manually using Excel files. Every month, the Care Houses have to send an invoice to the HFF which will pay the amount of money described in the invoice for the care of the patients (Step 3 in Figure 1.1). However, when a patient is taken to an external institution, the amount of days spent outside of the Care House should not be accounted for in the invoice, for example, if the patient enters the Care House on the first day of the month and has to be taken to an external institution for 5 days, then the amount of days accounted for in the invoice should be 25. This however, cannot be verified by the Hff. Once the HFF has access to the invoice the value described must be validated and, if correct, the HFF will pay for the cares of the patients given by the Care House (Step 4 in Figure 1.1). Finally, every year the financial staff of the HFF has to send an invoice to the CNSC detailing the total amount spent with every patient interned in the Care Houses (Step 5 in Figure 1.1).

1.3 Developed Platform

The project consists of a web application that can be accessed by the members of the HFF involved in the referral process and financial management, as well as the members of the Care Houses who manage the interned patients.

The platform is currently deployed in a server provided by the HFF and is being tested by the members of the hospital.

The access to the platform can be made using a public domain via `https://psm-hff.min-saude.pt/`, or using a private domain inside the HFF via `http://psm.hff.corp/`. The code of the platform is available in a GitHub repository, found in `https://github.com/AxelPrime/2021_2022_isel_tfm_psm`.

1.4 Outline

This document is divided in a total of 5 Chapters:

- Chapter 1 *Introduction* - This chapter describes the main goals for the creation of the project as well as a description of the current methods used by the HFF and

care house's staff in the process of creating and managing referrals and interments.

- Chapter 2 *Process Description* - This chapter consists of the definition of all the intervening actors, as well as their roles and responsibilities. This chapter presents a list of views that were created in order to serve as auxiliary tools to create the web pages in the final application.
- Chapter 3 *Technology* - This chapter contains a description of the technology used in the development of the project.
- Chapter 4 *Development* - In this chapter the different stages of the development of the project are described. This covers the creation of the database model, followed by the creation of the project and web application, and respective testing and deployment.
- Chapter 5 *Conclusions* - This chapter summarizes the main contributions of this project and presents future work that can be made in the development of the application.

2

Process Description

The following chapter contains the process description that users will have to follow in order to use the functionalities of the application. This chapter discusses the different types of users (actors) that are able to use the application, as well as their functions and types of actions that can be performed.

Currently, the process of sending and creating invoices is performed manually. The Care Houses must create an Excel file with the data of every patient present which indicates the start date of the invoice, the end date of the invoice, the total income associated with the patient and the total amount of money for the Typology (I, II and III). In this process, the Care House is responsible for counting the days that the patient has been taken care within the Care House. In case that the patient has to leave for consultation or internment in another hospital, the days that the patient spends outside of the Care House should not be accounted for in the invoice. This, however, is ignored in the current manual process, which leads to HFF's extra charging when the patient was outside of the Care House.

There are two types invoices that must be created:

- **Monthly Invoice** - This type of invoice is created by the Care Houses and is sent to the HFF on a monthly basis. At the end of every month, the Care House must compile a list of all the patients and send it to the HFF with the information regarding the number of days that each patient has been in the Care House and the associated receipt (this is given by the number of days the patient has been in the

care of the care house times the daily rate, e.g. 43€). There are three invoices that must be sent every month, one for each typology, along with the list of patients in the Care House. When the invoice is sent to the HFF, the financial department must validate if the amount of provided money in the invoice matches the amount of money described in the list of patients. After the validation, the HFF will then proceed to the payment of the invoice, either by self-paying or by asking the other institutions for the payment, in the case that the patient is referenced by another institution.

- **Yearly Invoice** - This type of invoice is created yearly by the HFF and must be sent to the CNSC. This invoice describes all the spending made by the Care Houses so that the CNSC may provide the financing. This invoice is only applicable to Typologies I and III. Each line of the invoice contains the information regarding a month of the patient spent in a Care House (e.g. if a patient spends 3 months in a Care House, there will be 3 lines for that patient). The lines of the invoice show the name of the patient, the process number, the date of start and end of the monthly invoice, the number of days invoiced, the card number of the patient and the patient was discharged.

2.1 Actors

After the identification of the application requirements, it was necessary to identify the different types of users that will interact with the various application's functionalities. Given the fact that there are multiple entities (HFF and Care Houses) involved in the process, there is a need to identify different types of users with different levels of access to information. Besides this, there are also different departments of the entities involved in the processes. After analyzing the facts indicated above, the identified actors are the following:

- **Institution Psychiatrist** - The Institution Psychiatrist is a member of HFF or another hospital that is responsible for the referral of patients to the Care Houses. This actor must insert the patient data inside the application, in order for the patient to be reviewed for approval.

- **Referral Reviewer** - After the referral of a patient and the evaluation of the process by the Care House Staff, this process has to be reviewed in order to determine if the patient is eligible to go to a Care House. This process is completed by the Referral Reviewer, which will analyze the patient data and approve or deny the

case. The Reviewer must indicate the reason for denial in the case that this is the decision.

- **Care House Staff** - The first responsibility of the Care House Staff is to approve the referral to a Care House. Once the referral is created, the Care House Staff will need to indicate if the Care House has vacancies to receive the patient. Once a patient has been approved to be sent to a Care House, the information that the patient has entered the Care House must be indicated in the application. To perform this action, the Care House Staff must indicate in the application that the patient has arrived. Besides this, the Care House Staff is also responsible for the indication of the patient leaving the care house for consultation in other facilities, as well as the return of the patient. In case of dismissal or patient's death, the Care House Staff must report the event in the application. The Care House Staff is also responsible for the invoicing of the patients present in the Care House. This process can be done by indicating the Typology of the patients to invoice, as well as the starting and end date of the invoice. The invoice will then be sent to the institution for analysis.

- **Financial Staff** - Once the invoice is sent to the HFF, the Financial Staff will analyze it to verify if there are inconsistencies. After the analysis, the invoice is either approved or denied. If approved, the invoice is processed, and the amount will be paid to the Care House.

- **Superuser** - The Superuser is a user that has all the permission of all the roles combined. This means that this particular user can view all the data of all the patients, currently in a Care house or awaiting the review of the referral, as well as make changes to the data (e.g. change the state of a patient in the Care House or approve a patient's referral). This user also has the ability to revert certain actions made by other users, for example, if a Care House Staff user mistakenly discharges a patient, the Superuser has the ability to overturn this action.

2.2 Roles, Responsibilities and Actions

With the definition of the actors and the basic responsibilities for each one, it is now necessary to define the actions and roles each actor will be able to perform in the application. In the following sections it will be defined what actions are available for each actor, as well as the responsibilities and roles. In order to aid in the definition of the actions, a set of mock-ups[1] was created. This will allow the HFF to visualize the idea

for the application, giving them the opportunity to provide feedback on the potential views that will be created.

2.2.1 Institution Psychiatrist

The Institution Psychiatrist is the member of the HFF that is responsible for the referral of patients that transit to the Care Houses. The psychiatrist is responsible for choosing the Care House that best fits the patient's needs. In order to refer the patient, the psychiatrist must indicate the information relative to the patient (Figure 2.3) such as his name, SNS number, Care House to send the patient to, the type of mental disease and the birth date of the patient. The Institution Psychiatrist also needs to attach a file that contains the medical information regarding the patient, along with the cares and medication. After the referral is completed, the psychiatrist will have access to the current state of this patient (Figures 2.1 and 2.2), indicating if the patient has been approved to be sent to the Care House or in alternative, the request is denied. The psychiatrist must also be able to verify the state of every patient that has been referred. There are a total of 5 states for the patient's referral (Figure 2.4): "*Aprovado*" ("Approved"), "*Rejeitado*" ("Rejected") and "*Aguarda Resposta da Casa de Saúde*" ("Awaits Care House's Response"), "*Aguarda Vaga*" ("Awaits Opening") and "*Aguarda Resposta do HFF*" ("Awaits HFF's Response").

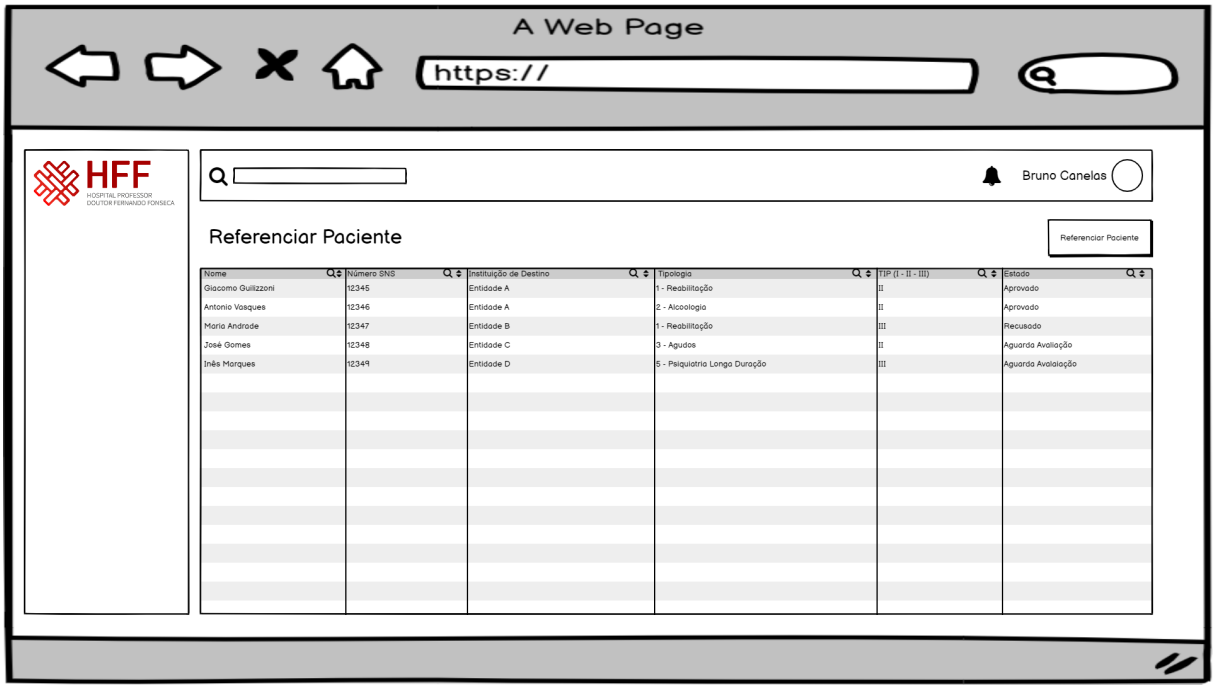

Figure 2.1: List of referenced patients

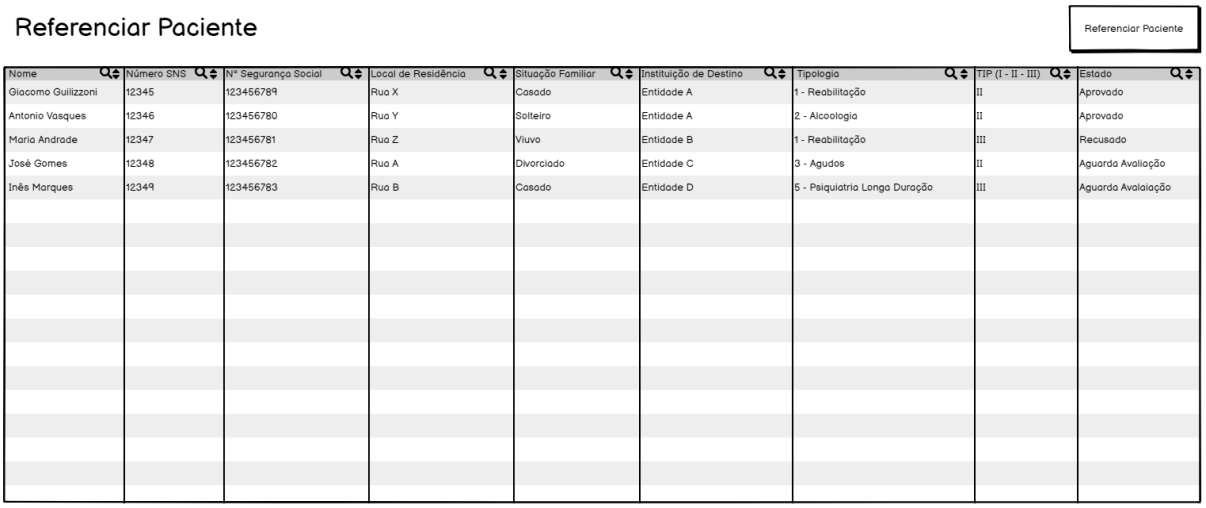

Figure 2.2: List of referenced patients (zoomed)

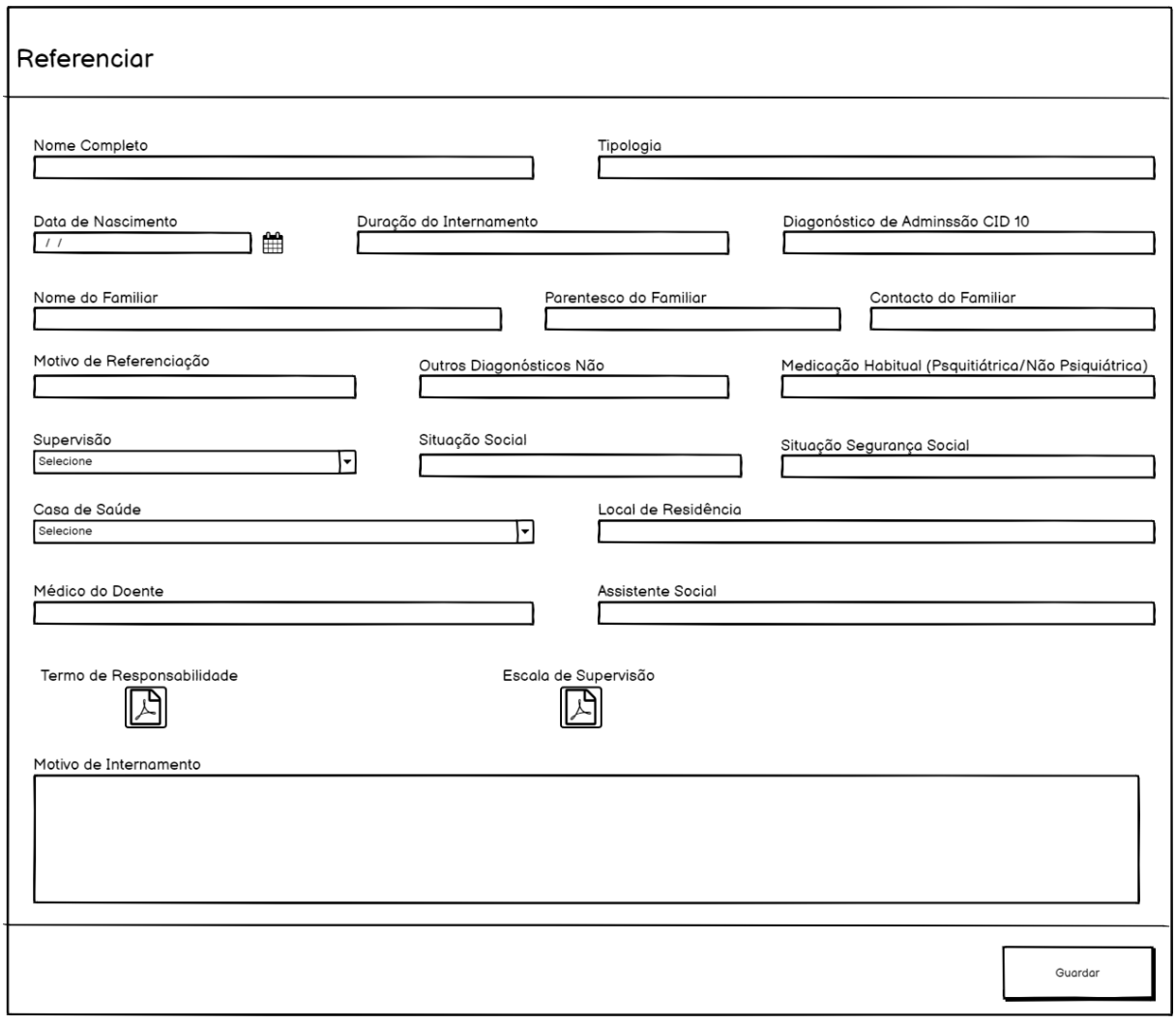

Figure 2.3: Required fields to refer a patient

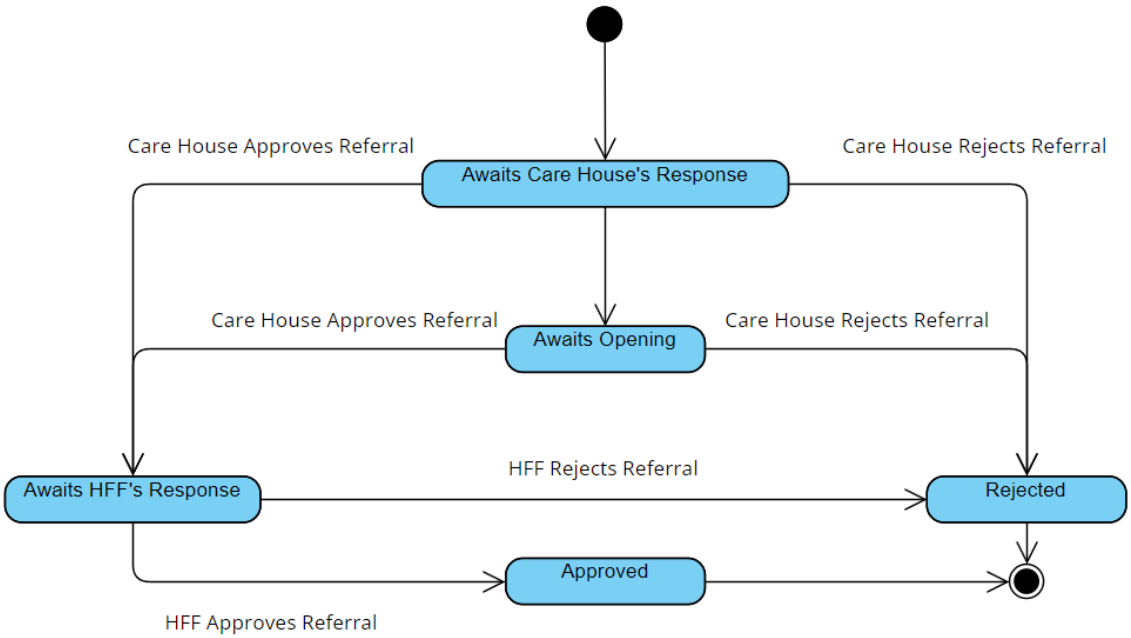

Figure 2.4: Possible states of a referral

2.2.2 Referral Reviewer

When a patient is referred by the Institution Psychiatrist and approved to go to the Care House by the Care House Staff, the information must be reviewed by the Referral Reviewer. The reviewer must analyze the data introduced by the Institution Psychiatrist in order to approve or deny the request to send the patient to a Care House (Figures 2.6 and 2.7). Once a decision has been made, the reviewer must indicate if the patient has been approved or denied. For this, the reviewer has access to the list of currently open cases (Figure 2.5) that they can choose from, in order to indicate the decision. This view is similar to the one represented in Figure 2.1, however there is the possibility to select multiple patients to evaluate. After the decision is made, the reviewer will have access to the list of previous decisions, where they will also find the case that was just reviewed.

Figure 2.5: List of patients to approve/deny

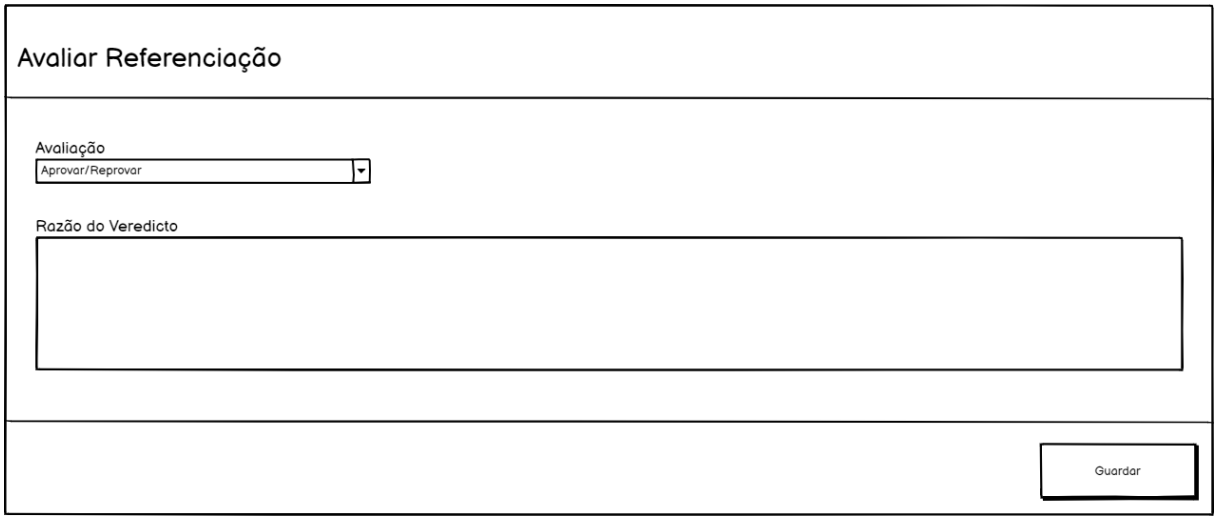

Figure 2.6: Required fields to approve/deny a referral

Figure 2.7: Approve/Deny Confirmation

2.2.3 Care House Staff

Once the patient has been referred by the Institution Psychiatrist, the process is sent to the Care House in order to be determined if the Care House is capable of taking care of the patient and if there are vacancies available to receive the patient. In case of acceptance, the process will then be evaluated by the referral reviewer. These views are similar to the ones displayed in Figures 2.5 and 2.6, however, in this case the used has an extra option in the evaluation that indicates that the patient needs to await an opening in the Care House.

After the approval, the patient will be sent to the Care House and taken care of. The Care House staff will be responsible for the patient and needs to account the days that the patient is in the care of the Care House. In order for this to happen, the Care House staff will need access to the list of patients present in the Care House (Figures 2.8 and 2.9). The Care House staff will also need the ability to indicate when a patient enters the Care House, as well as when the patient leaves the Care House, either for consultation in another facility or medical discharge (Figure 2.10). Once the patient arrives the Care House, the Staff will need to indicate in the application that the patient has entered the Care House. For this, the Care House Staff will have access to the list of patients that have been approved to go to the Care House but have not been received yet. When the patient is received, the Care House Staff will need to go to the list and confirm the

reception of the patient. Once the reception is confirmed the patient will be moved to the list of patients currently in the care of the Care House.

In case of need to take the patient to another institution, the Care House Staff will need to indicate in the application that the patient is leaving, indicating the reason why this action is happening and the facility that will receive the patient temporarily. The patient will then appear on a list of patients currently in temporary care of another institution. Once the patient returns, the Care House Staff will need to indicate in the application that the patient has reentered the Care House, sending the patient back to the list of patients currently in the care of the Care House. In the case of a medical discharge or death, the Care House staff will also have the ability to report the event in the application.

There are a total of 7 states for the patients in the Care House (Figure 2.11): "*Internado*" ("Hospitalized"), "*Consulta Externa*" ("External Consultation"), "*Agarda Entrada*" ("Awaiting Entry"), "*Alta Médica*" ("Medical Discharge"), "*Alta sem Parecer*" ("Self Discharge"), "*Transferência*" ("Transfer") and "*Falecido*" ("Death").

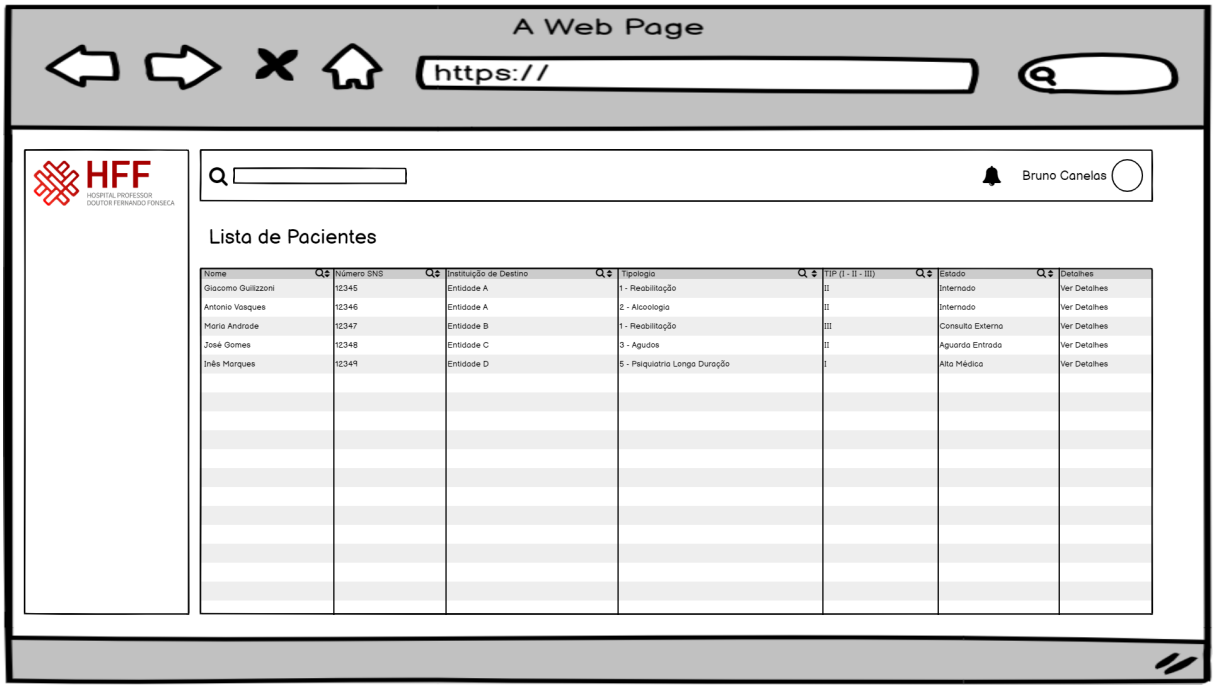

Figure 2.8: List of patients of the Care House

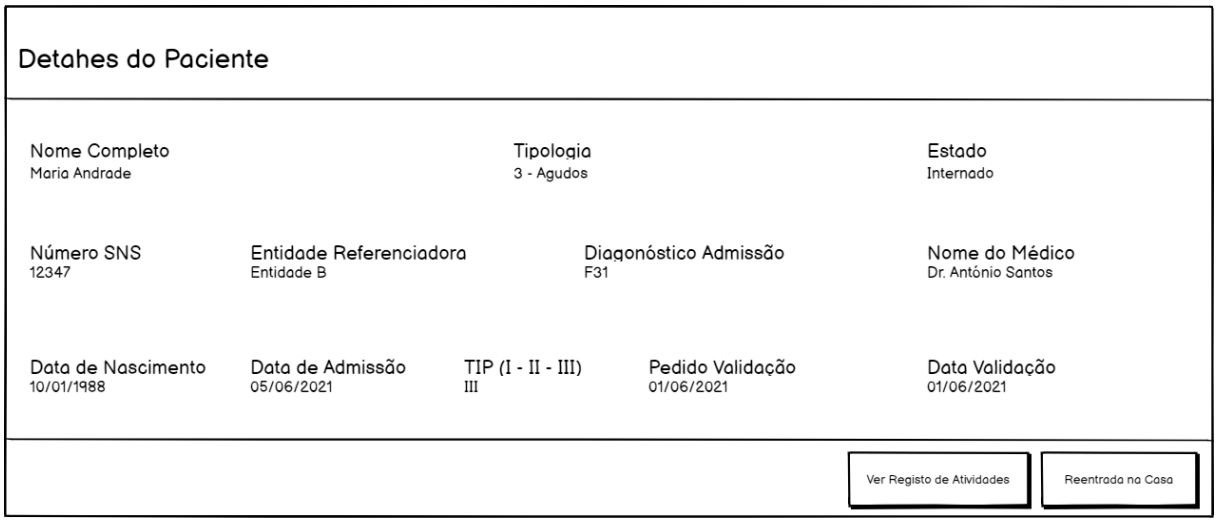

Figure 2.9: Patient details

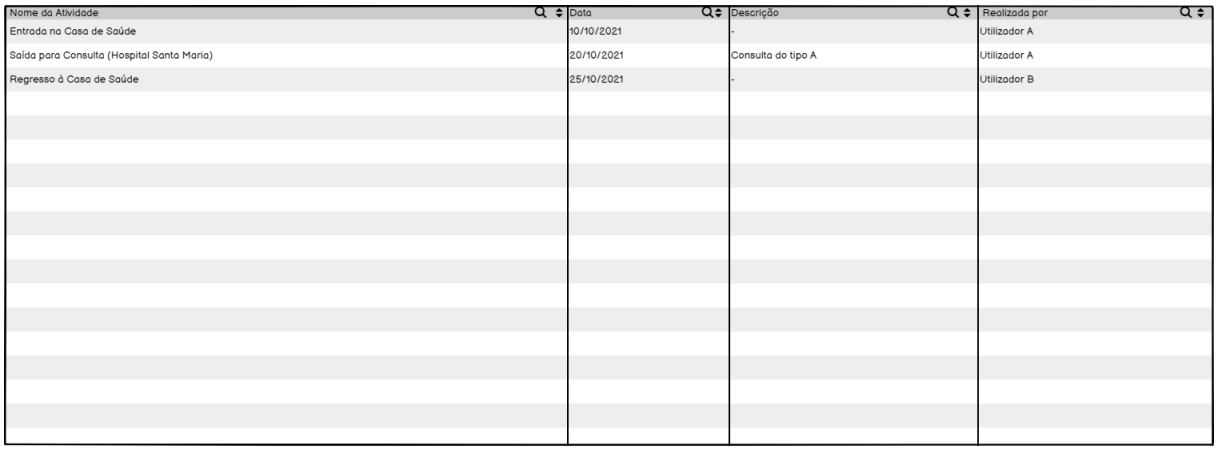

Figure 2.10: Patient Activity Log

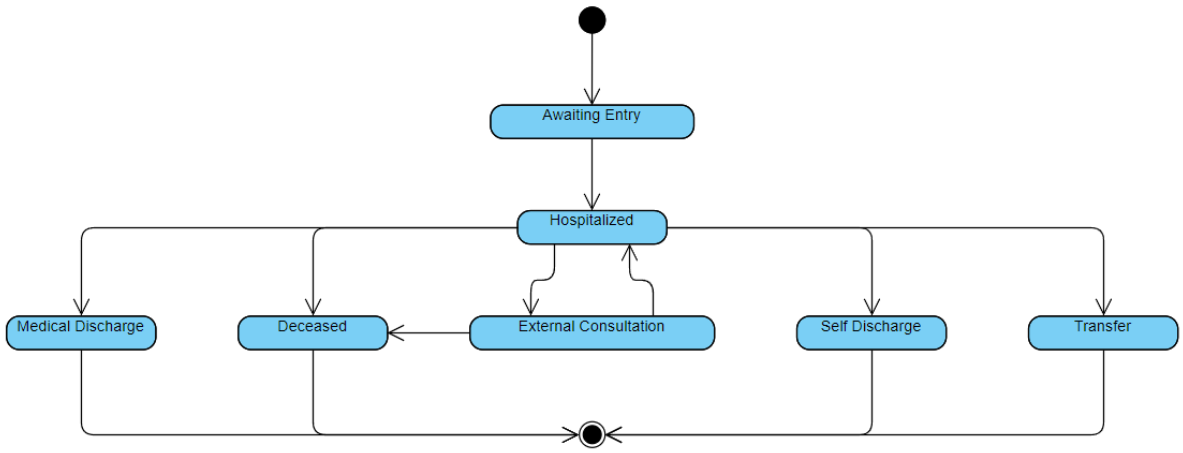

Figure 2.11: Patient States transition diagram

Besides the patient management, the Care House Staff will also have access to the invoicing of patients. At the end of every month the Care House Staff will need to generate the receipts for the care of the patients. In the invoicing area, the Care House Staff will have the ability to generate the necessary receipts and send them to the Financial Staff of the HFF (Figure 2.12). When the Care House Staff needs to generate the receipts, the application will automatically generate the list of patients for preview (Figure 2.13) as well as one receipt for each typology. After the verification of the data, the Care House Staff will need to indicate that the information is correct. After the validation the documents will be sent to the Financial Staff automatically.

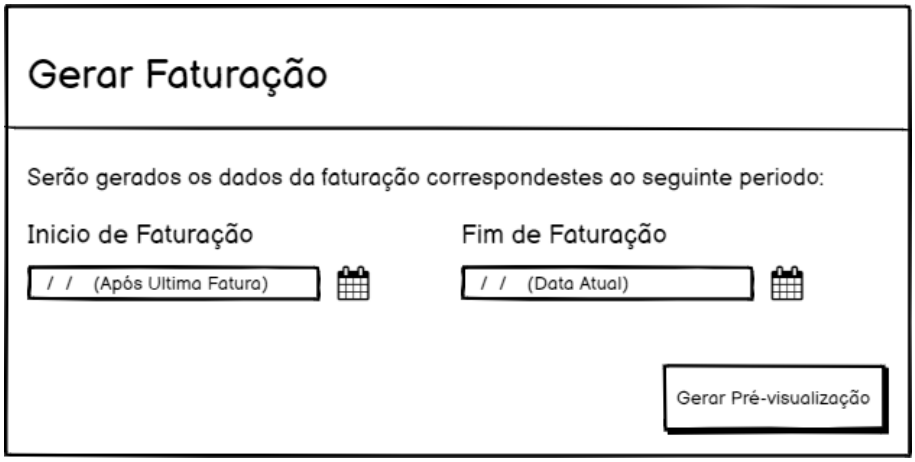

Figure 2.12: Invoice creation

Nome	Número SNS	Inicio Faturação	Fim Faturação^v	Detalhes	Valor a Faturar
Giacomo Guilizzoni	12345	01-10-2021	31-10-2021	Ver Detalhes	1333.00€
Antonio Vasques	12346	01-10-2021	31-10-2021	Ver Detalhes	1333.00€
Maria Andrade	12347	06-10-2021	31-10-2021	Ver Detalhes	1333.00€
José Gomes	12348	10-10-2021	31-10-2021	Ver Detalhes	1333.00€
Inês Marques	12349	10-10-2021	31-10-2021	Ver Detalhes	1333.00€
Total					xxxxx€

Figure 2.13: Invoice Preview

2.2.4 Financial Staff

Once the Care House Staff has sent the invoices, the Financial Staff will need to verify the values present in the receipts. For this, the Financial Staff will need access to the

list of invoices that are currently awaiting approval for each one of the Care Houses (Figure 2.14). For each invoice, the Financial Staff will have access to the documents regarding the receipts, as well as the list detailing the individual values invoiced for each patient (Figure 2.15). The Financial Staff will then be able to cross reference the values to make sure that values in the receipts are correct. Once the verification is complete, the Financial Staff will then be able to approve the payment of the invoice, altering the state of that invoice. In case of rejection of the invoice, the Care House will need to send a new invoice. The Financial Staff will also have access to the previous invoices generated by the Care Houses. Finally, the Financial Staff will also be able to generate the yearly invoice to send to the CNSC. The application will automatically generate the invoice, showing a preview of the data to the Financial Staff. If approved, the invoice will then the sent to the CNSC.

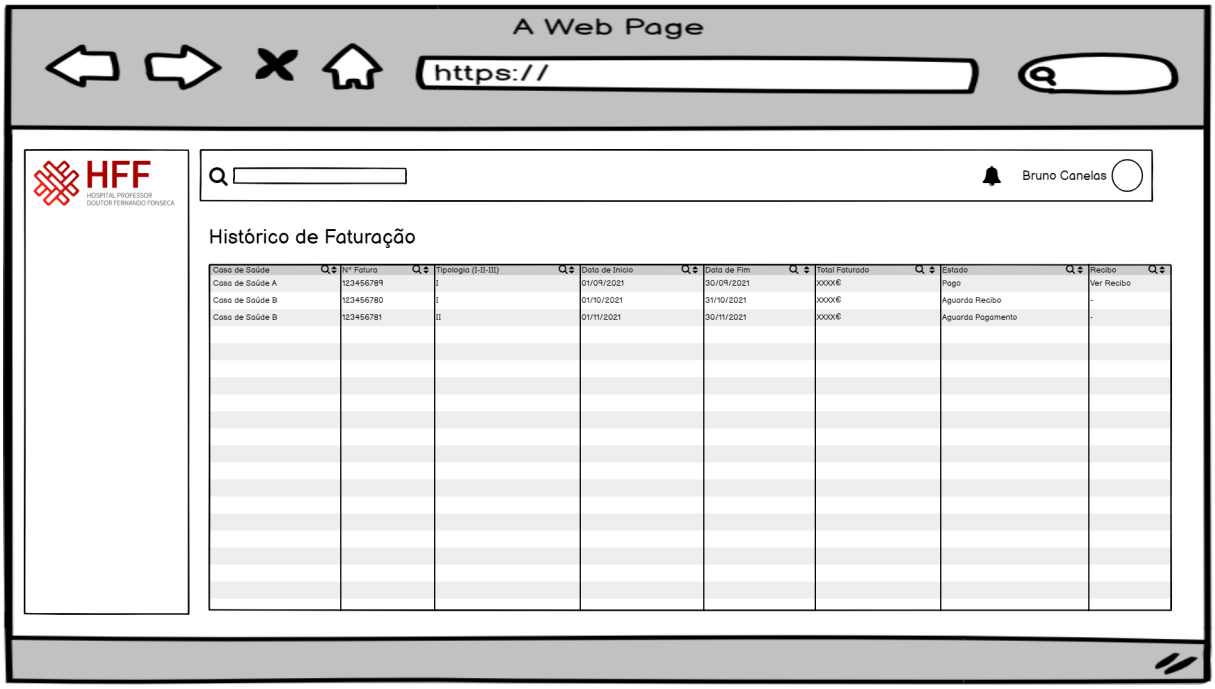

Figure 2.14: Invoice history

Nome	Número SNS	Inicio Faturação	Fim Faturação^v	Detalhes	Valor a Faturar
Giacomo Guilizzoni	12345	01-10-2021	31-10-2021	Ver Detalhes	1333.00€
Antonio Vasques	12346	01-10-2021	31-10-2021	Ver Detalhes	1333.00€
Maria Andrade	12347	06-10-2021	31-10-2021	Ver Detalhes	1333.00€
José Gomes	12348	10-10-2021	31-10-2021	Ver Detalhes	1333.00€
Inês Marques	12349	10-10-2021	31-10-2021	Ver Detalhes	1333.00€
Total					XXXXX€

Figure 2.15: Invoice Details

The Financial Staff is also responsible for sending the invoices to the External Institutions that have sent their patients to one of the HFF Care Houses. In order to do this, the Financial Staff will have access to a list containing the information of the number of patients from each External Institution per month (Figure 2.16). This list provides the total amount of money spent in the daily care of the patients from all institutions. The Financial Staff is also able to check details for each month (Figure 2.17) in order to determine the money that each individual External Institution will have to pay.

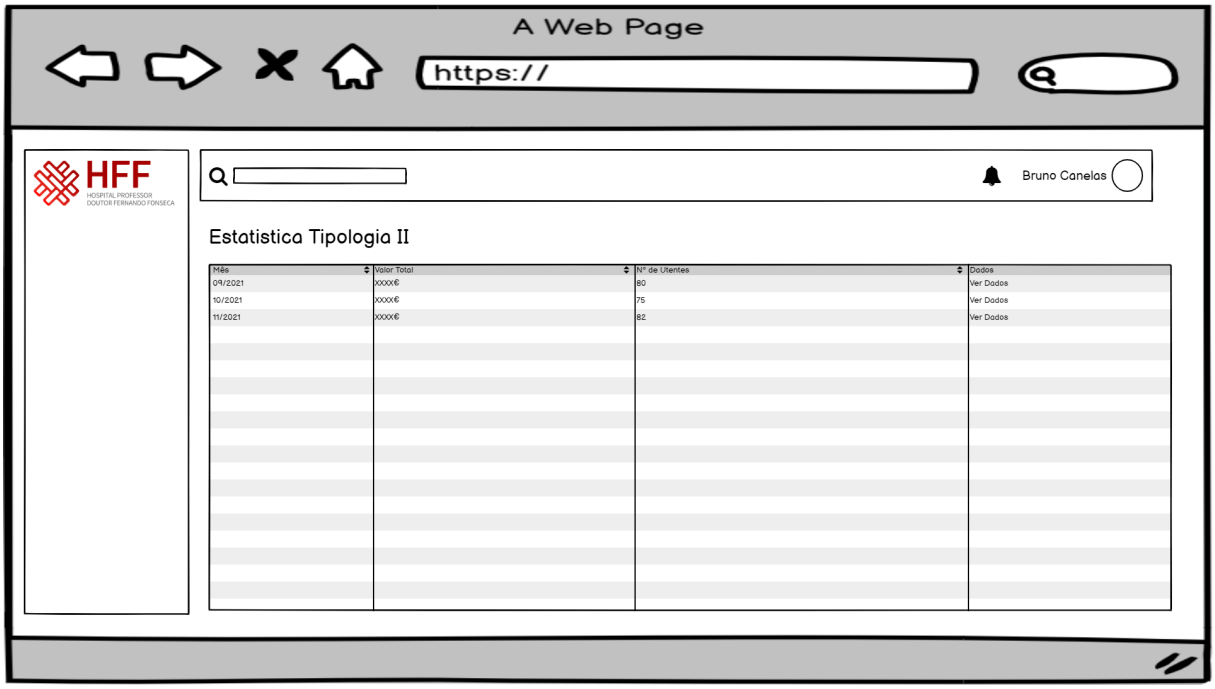

Figure 2.16: Typology II Statistics

Mês	Soc. Parc.	NIF	Nome Insituição	Valor	Nº de Utentes
09/2021	1000	123456789	Intituição A	XXXX€	10
09/2021	1100	123456780	Intituição B	XXXX€	20
09/2021	1200	123456781	Intituição C	XXXX€	15
09/2021	1300	123456782	Intituição D	XXXX€	5
09/2021	1400	123456783	Intituição E	XXXX€	30
Total				XXXX€	80

Figure 2.17: Typology II Month Details

The Financial Staff will need access to the information regarding the number of patients of all Typologies in all the Care Houses in order to verify if the number of patients has changed through the year. For this, the Financial Staff will have access to tables containing the number of patients divided into the corresponding Typology and Care Houses per month, as well as the number of days invoiced on each month (Figure 2.18).

Tipologia	Unidades Prestadoras	janeiro			fevereiro			março			abril			maio		
		Nº Doentes	Diárias	Diárias (€)	Nº Doentes	Diárias	Diárias (€)	Nº Doentes	Diárias	Diárias (€)	Nº Doentes	Diárias	Diárias (€)	Nº Doentes	Diárias	Diárias (€)
Doentes Internados até 31 Dez 2013 (Tip I)	Casa Saúde de Santa Rosa de Lima	20	614	26 406	20	556	23 908	20	619	26 617	20	586	25 198	18	558	23 994
	Casa Saúde do Telhal	234	7 146	307 287	229	6 238	268 234	226	6 914	297 302	227	6 645	285 735	222	6 737	289 691
	Casa Saúde da Idanha	261	7 967	342 581	256	7 064	303 752	254	7 817	336 131	250	7 476	321 468	250	7 740	332 820
	Subtotal	515	15 727	676 274	505	13 858	595 894	500	15 350	660 050	497	14 707	632 401	490	15 035	646 505
Doentes a refaturar pelo HFF (Tip II)	Casa Saúde de Santa Rosa de Lima	2	62	2 666	2	56	2 408	2	62	2 666	2	60	2 580	2	62	2 666
	Casa Saúde do Telhal	51	1 518	65 274	50	1 258	54 094	49	1 454	62 522	52	1 455	62 565	56	1 591	68 413
	Casa Saúde da Idanha	19	575	24 725	18	504	21 672	19	572	24 596	23	692	29 756	23	712	30 616
	Subtotal	73	2 185	92 665	70	1 818	78 174	70	2 088	89 784	77	2 207	94 901	81	2 365	101 695
Doentes referenciados/ área intervenção HFF+HPPP (Tip	Casa Saúde de Santa Rosa de Lima	34	1 028	44 204	34	905	38 915	33	939	40 377	33	958	41 194	34	1 045	44 935
	Casa Saúde do Telhal	58	1 741	74 863	58	1 574	67 682	65	1 909	82 087	64	1 901	81 743	71	2 035	87 505
	Casa Saúde da Idanha	52	1 571	67 553	53	1 416	60 888	61	1 786	76 798	63	1 882	80 926	65	1 973	84 839
	Subtotal	142	4 279	183 997	145	3 895	167 485	159	4 634	199 262	160	4 741	203 863	170	5 053	217 279

Casas de Saúde	janeiro			fevereiro			março			abril			maio		
	Nº Doentes	Diárias	Diárias (€)	Nº Doentes	Diárias	Diárias (€)	Nº Doentes	Diárias	Diárias (€)	Nº Doentes	Diárias	Diárias (€)	Nº Doentes	Diárias	Diárias (€)
CSSRL Total (Tip I+II+III)	56	1 704	73 272	56	1 517	65 231	55	1 620	69 660	55	1 604	68 972	54	1 665	70 434
CST Total (Tip I+II+III)	343	10 405	447 424	337	9 070	390 010	340	10 277	441 911	343	10 001	430 043	349	10 363	445 609
CSI Total (Tip I+II+III)	331	10 082	433 526	327	8 984	386 312	334	10 175	437 525	336	10 050	432 150	338	10 425	448 275
Total - Tip I+II+III	730	22 191	954 226	720	19 571	841 553	729	22 072	949 096	734	21 655	931 165	741	22 453	964 318
CSSRL (Tip I+III - CP - HFF)	54	1 642	70 610	54	1 461	62 823	53	1 558	66 994	53	1 544	66 392	52	1 603	68 929
CST (Tip I+III - CP - HFF)	291	8 856	380 817	287	7 812	335 916	291	8 823	379 389	291	8 546	367 478	293	8 772	377 196
CSI (Tip I+III - CP - HFF)	312	9 508	408 844	309	8 480	364 640	315	9 603	412 929	313	9 358	402 394	315	9 713	417 659
Subtotal - Tip I+III- CP - HFF	657	20 006	860 271	650	17 753	763 379	659	19 984	859 312	657	19 448	836 264	660	20 088	863 784

Figure 2.18: Patients Monthly Details

In order to compile the yearly invoice, the Financial Staff will need access to the data regarding the information necessary to build the invoice. The Financial Staff will be able to generate two tables regarding the yearly invoice: one table will contain the data of the invoice (Figure 2.19) and the other one will contain a summary of all the patients in the Care Houses (Figure 2.20). After generating theses tables they will be available to be downloaded in XLSX format in order to be edited by the Financial Staff if necessary.

Nome	Data Nascimento	Nº Cartão	Nº Beneficiário	Nº Processo	Data Inicio Fatura	Data Fim Fatura	Dias Faturados	Valor Total	Código Alta
José Pinto	01/01/1960	123456789	123456789	12345	01/01/2021	31/01/2021	31	XXXXX€	-
José Pinto	01/01/1960	123456789	123456789	12345	01/02/2021	28/01/2021	28	XXXXX€	-
José Pinto	01/01/1960	123456789	123456789	12345	01/03/2021	31/03/2021	31	XXXXX€	-
Andreia Santos	10/06/1986	123456780	123456780	12346	01/01/2021	31/01/2021	31	XXXXX€	-
Andreia Santos	10/06/1986	123456780	123456780	12346	01/02/2021	15/02/2021	15	XXXXX€	1
Pedro Gonçalves	25/12/1970	123456781	123456781	12347	01/01/2021	20/01/2021	20	XXXXX€	20

Figure 2.19: Yearly Invoice Details

Nome	Nº Cartão	Data Inicio Faturação	Data Fim Faturação	Código Intituição Destino
José Pinto	123456789	01/01/2021	-	03
Andreia Santos	123456780	01/01/2021	15/02/2021	04
Pedro Gonçalves	123456781	01/01/2021	20/01/2021	08

Figure 2.20: Yearly Invoice Summary

3

Technology and Architecture

The following chapter details the technologies used in the development of the application. This chapter contains information regarding the technological environment, such as: the managed runtime environment, the development framework and the database management system used to store the data necessary for the correct functioning of the application.

3.1 Managed Runtime Environment

The development of the back-end side of the application is supported by the Python[2][3] development and runtime environment. Python is a general-purpose, interpreted, object-oriented high-level programming language that is easy to use and develop. This means that Python can be used in a wide range of different types of applications, while resembling natural language. Being an interpreted language means that Python code needs to be translated to an intermediate format (bytecode) which will then run on a virtual machine (Python Virtual Machine) that serves as an interpreter that translates bytecode into machine code.

The Python programming language is focused on code readability which makes it easy to use, develop and learn. Besides this, the Python language is also open source, meaning that anyone can download python and start developing applications with ease, and with a big community of programmers it is also not hard to find helpful tips. Being a general-purpose language means that Python can be used to develop various types of

applications, which means that there is a large number of available libraries that can be used to develop these applications.

However, Python is not a perfect language and with that come some disadvantages. The Python language has some limitations with threading. Being a single-threaded language makes it difficult to develop multi-threaded applications due to the fact that Python uses a Global Interpreter Lock (GIL), that is a mutex that only allows one thread to execute at a time.

3.2 Framework

Using Python as a programming language allows the use of the Django Framework. Django is an open-source framework written in Python composed of a set of components that help the development of websites. The Django framework is versatile and allows the developed application to deliver contents in various formats (e.g. HTML or JSON). Django is also a secure framework that helps developers avoid some common security mistakes. The framework provides a set of default tools and functionalities to achieve this purpose, like storing a unique identifier of the user session on a cookie while actual data is stored in the database. Django also provides some security regarding vulnerabilities like SQL Injection, Cross-Site Scripting (XSS) and Cross-Site Reference Forgery (CSRF).

The Django Framework allows the use of a diverse number of Database Management Systems, like PostgreSQL or MySQL. The framework is designed with a Code First mentality. The developer creates a set of Models that represent the database tables. In these models the developer can indicate the fields of each table, including the type, name, constraints and other attributes. Once the models are defined or updated the developer needs to reflect these changes in the database. For this, the Django framework creates a set of migrations that contain the information regarding table creation and updates.

Using the Django framework allows the developer to access an administration page accessible only by administrative users. This page allows the user to interact with the data present in the database in a user-friendly interface. The user can view, create, update and delete entries on the database tables, and, in a more advanced format, create functions to execute certain tasks on the data.

3.3 Database

For the database management system, the application uses a PostgreSQL[4] database. PostgreSQL is an advanced, open-source relational database management system that supports both SQL and JSON for relational and non-relational querying, respectively.

The PostgreSQL project started in 1986 and was originally designed to be executed in UNIX-like systems. Today, PostgreSQL is supported by many operating systems including Windows and MacOS.

PostgreSQL is a highly stable database that is backed by an open source community and it is primarily used in web applications as well as analytical and mobile applications.

The PostgreSQL contains a variety the features that are also offered by other enterprise-class database management systems such as:

- user-defined types
- table inheritance
- nested transactions
- multi-version concurrency control
- asynchronous replication

In more recent versions, there was also added support for features like Tablespaces and Point-in-Time Recovery.

The use o PostgreSQL comes with a set of advantages like:

- The PostgreSQL Project is open-source, meaning that there is no need to pay for features. All the features of the management system are available to use for free
- It is compliant with the SQL Standard
- The possibility to process complex data types, for example Geographical data
- It allows the user to create functions, triggers, data types, etc.
- Supports JSON structures
- It's a cross platform system, meaning that it can be executed in a wide range of operating systems

Despite all these advantages, the use of the PostgreSQL management system also comes with some disadvantages:

- The PostgreSQL management system is not available on all hosts by default
- Read queries are slower than other management systems, for example MySQL
- PostgreSQL focuses is compatibility, so improvements to performance take more time to develop when compared to other systems

3.4 Deployment

In order to deploy the platform, the application will be executed on an Ubuntu 20.04 LTS[5] operating system using Gunicorn[6] and Nginx[7] in order to serve the application resources such as API's and static files like JavaScript files and CSS files.

The Ubuntu operating system is an open-source Linux distribution based on the Debian operating system. Being an open-source operating system means that Ubuntu can be used both by individual users or enterprises without the need to pay for software licenses or the use of specific devices. Ubuntu can be installed in a wide range of devices including personal computers, servers, IoT devices and robots, given the fact that the hardware requirements necessary to run this operating system are very low-end, requiring only a 1GHz processor, 1GB of RAM and 2.5GB of storage space. This makes Ubuntu popular among companies like Google and NASA who use Ubuntu on their servers. The 20.04 LTS version of Ubuntu is the latest long term support version of Ubuntu released (April 23rd 2020) and will receive standard updates until April 2025 and will receive security updates until April 2030.

In order to deploy a Django application on a web server it is necessary to use a communication interface capable of translating the Python language. Currently, Django supports two interfaces:

- **WSGI**[8] - Stands for Web Server Gateway Interface. WSGI is a specification that describes how web servers and web applications communicate with each other, and also how web applications can be chained together to process one request. This is the main standard used in the communication between web servers and applications, however it only supports synchronous code.
- **ASGI**[9] - Stands for Asynchronous Server Gateway Interface. ASGI is a successor to the WSGI specification and provides a standard interface that can be used by

asynchronous-capable Python web servers, frameworks, and other types of applications. This is a new and asynchronous friendly standard that allows Django sites to use asynchronous features, both for Python itself and Django.

Given these two possible interfaces, it was decided to use ASGI given the fact that it supports asynchronous code. In order to use this interface, it is necessary to use Gunicorn in conjunction with Uvicorn workers.

Gunicorn is WSGI application server that serves as the communication bridge between the HTTP server itself and the application. The HTTP server is in charge of accepting requests and handling HTTP connections. The requests that are meant for the applications are then passed through Gunicorn to the application. The response to the request is then received by Gunicorn which transmits it back to the client through the HTTP server. Given the fact Python is a single-threaded language, Gunicorn allows the use of multiple synchronous worker processes to communicate with the application. In order to use the ASGI asynchronous capabilities, there is the need to use Uvicorn workers. Uvicorn is an ASGI application server implementation that allows Gunicorn to run asynchronous workers. Combining Gunicorn with the Uvicorn workers allows the combination of asynchronous workers with the process management present in Gunicorn.

Finally, Nginx will be used as the HTTP server, which has the responsibility of handling the requests received, serving static files like JavaScript and CSS files and managing the domain logic.

Nginx is an open-source HTTP server that can also function as a proxy server, as well as a reverse proxy and a load balance for other servers. The use of Nginx provides some benefits like:

- A friendly configuration and modern designed
- Allows the handle of multiple connections
- Good compatibility with Python web applications
- Capable of handling thousands of concurrent connections

However, Nginx also suffers from a small community when compared to other web servers like Apache, and it does not offer the use of many modules and extensions.

In order to deploy the application in a production environment, the Nginx Web Server was configured in order to be able to accept requests from two domains, namely `https:`

//psm-hff.min-saude.pt/ and http://psm.hff.corp/, redirecting the requests to the same Gunicorn services, as well as a second instance of the application used only for testing and available only in the HFF's internal network via http://psm.hff.corp:8080/, using a separate Gunicorn service. The two deployed instances of the platform use different databases and, in the case of the testing platform, a secondary database used to simulate the Hosix WebService.

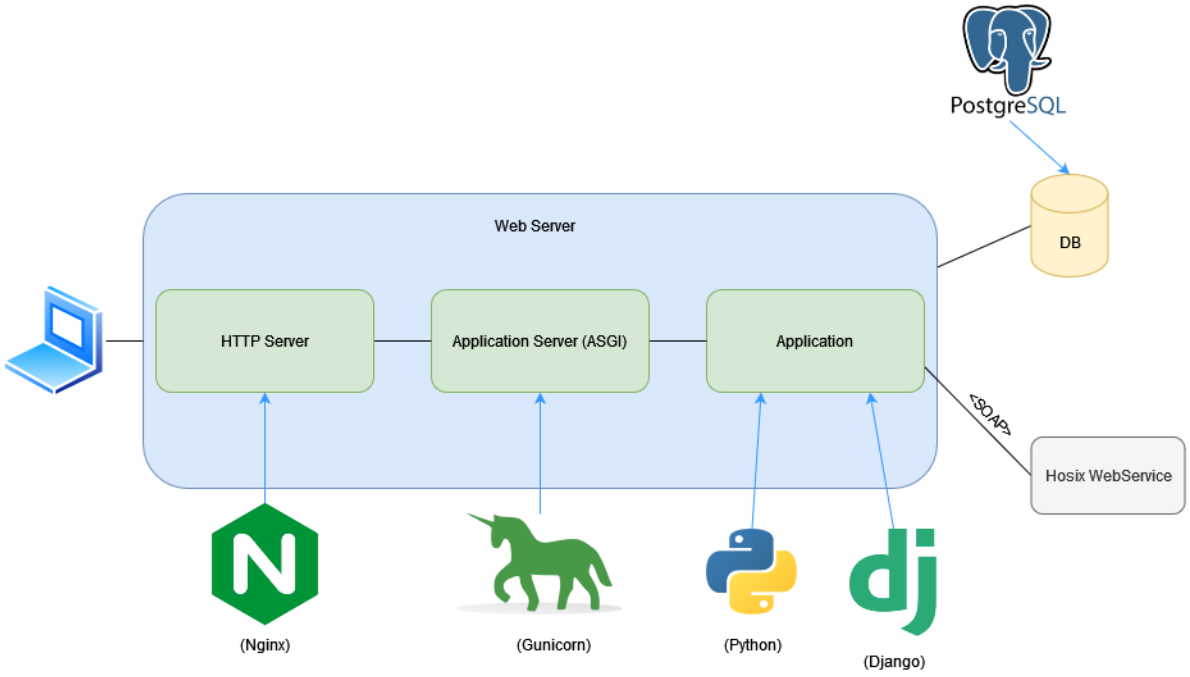

Figure 3.1: Platform Architecture

Given the tools used in creation and deployment of the platform, Figure 3.1 is a representation of the platform's architecture. The architecture serves as a demonstration for how the different tools interact in order for the platform to be used by the end user.

3.5 Other Tools

In order to integrate the application with the HFF database, the members of the hospital have provided a WebService that allows the application to interact with the Hosix system, where the HFF's patient data is stored. The documentation for this WebService can be verified in Appendix A.

The communication between the application and the Hosix WebService is done via the SOAP[10] (Simple Object Access Protocol) protocol. The protocol is a specification

for the exchange of structured information, typically used in the implementation of web services. The messages exchanged are written using XML (Extensible Markup Language) and allows clients to make requests to a web service and receive a response, independently of language and platform.

3.6 Installation Guide

In order to run an instance of the PSM application, a series of steps must be followed in order to ensure that the application is correctly configured. This guide will allow any user to configure the execution environment in order to test the application and all its functionalities.

3.6.1 Step 1 - Configuring the Database

The PSM application relies on a PostgreSQL database to store data regarding users, internments, referrals and other useful information. As such, the first step is to create a database using the command below:

```
1 CREATE DATABASE psm_database;
```

Listing 3.1: Create Database

In order to use the database, a user must be created with the necessary permissions to make queries over the data. The user can be created by executing the commands below:

```
1 CREATE USER myprojectuser WITH PASSWORD 'password';
2 GRANT ALL PRIVILEGES ON DATABASE psm_database TO myprojectuser;
```

Listing 3.2: Create Database User

3.6.2 Step 2 - Setting up the Python Environment

After the database has been created with success, the next step is to create the environment to execute the application. As such, the command below will create the Python virtual environment where the Python package requirements will be installed:

```
1 cd psm/
2 python3 -venv env
3 source env/bin/activate
```

Listing 3.3: Create Environment

After the creation of the environment, it is necessary to install the Python package requirements. The command below will complete the installation:

```
pip install -r requirements.txt
```

Listing 3.4: Install Requirements

Finally, in order to create the connection between the application and the database, it is necessary to create a .env file with the following variables:

```
# Django Secret Key
SECRET_KEY=key

# Prod DB Data
PROD_DB_USER=myprojectuser
PROD_DB_NAME=psm_database
PROD_DB_PASSWORD=password

# Dev DB Data
DEV_DB_USER=myprojectuser
DEV_DB_NAME=psm_database
DEV_DB_PASSWORD=password
```

Listing 3.5: .env Variables

3.6.3 Step 3 - Inserting the test data

The application can only be executed using the development settings, since the access to the Hosix WebService is only allowed in the HFF's servers. As such, the next commands will be executed using the --settings=psm_django.settings.dev flag.

The first step in the data insertion is to create the tables in the database, which are defined in the models of each application. The tables will be created using the migrations previously created using the following command:

```
python manage.py migrate --settings=psm_django.settings.dev
```

Listing 3.6: Migrate

After applying the migrations, the test data is now ready to be inserted. A script will be run that will create a set of institutions, care houses, patients, users, referrals, and

other records that can be used to fully test the application. The insertion of the data can be done by running the command listed below:

```
python manage.py runscript create_test_data --settings=psm_django.settings.dev
```

Listing 3.7: Create Test Data

In order to simulate requests to the Hosix WebService, it is necessary to create an SQLite database. The database can be created using the following command:

```
python manage.py runscript create_local_patient_db --settings=psm_django.settings.dev
```

Listing 3.8: Create Patient Test Database

3.6.4 Step 4 - Running the application

After completing the previous steps successfully, the application is ready to be executed. To do this, execute the command listed below. The command will run the application on host 0.0.0.0 and port 8000:

```
python manage.py runscript runserver 0.0.0.0:8000 --settings=psm_django.settings.dev
```

Listing 3.9: Run the Application

The execution can be terminated using Ctrl + C.

4

Development

The following chapter refers to the various phases of the project's development. This chapter features the documentation for the Database Model, the folder structure for the Django Project and the description of the pages used to interact with the application.

4.1 Database Model

In order to better structure the application and data needed to be stored persistently, the development of the project started with the definition of the Database Model (Entity-Relationship model) that is used by the back-end. Given the requirements provided by the HFF, the model was created and organized in different sections, each one necessary for a better readability and general comprehension of the data being stored. The different sections are the following: **User Management**, **Entities**, **Internment Management** and **Financial**. During this chapter, there will be a detailed explanation of each of these sections, as well as the tables created for each one.

4.1.1 User Management

This model section refers to the tables and stored data that is associated with the users. In here, we can find components like the email and name of the users, as well as the notifications used to inform the users of the need for certain actions.

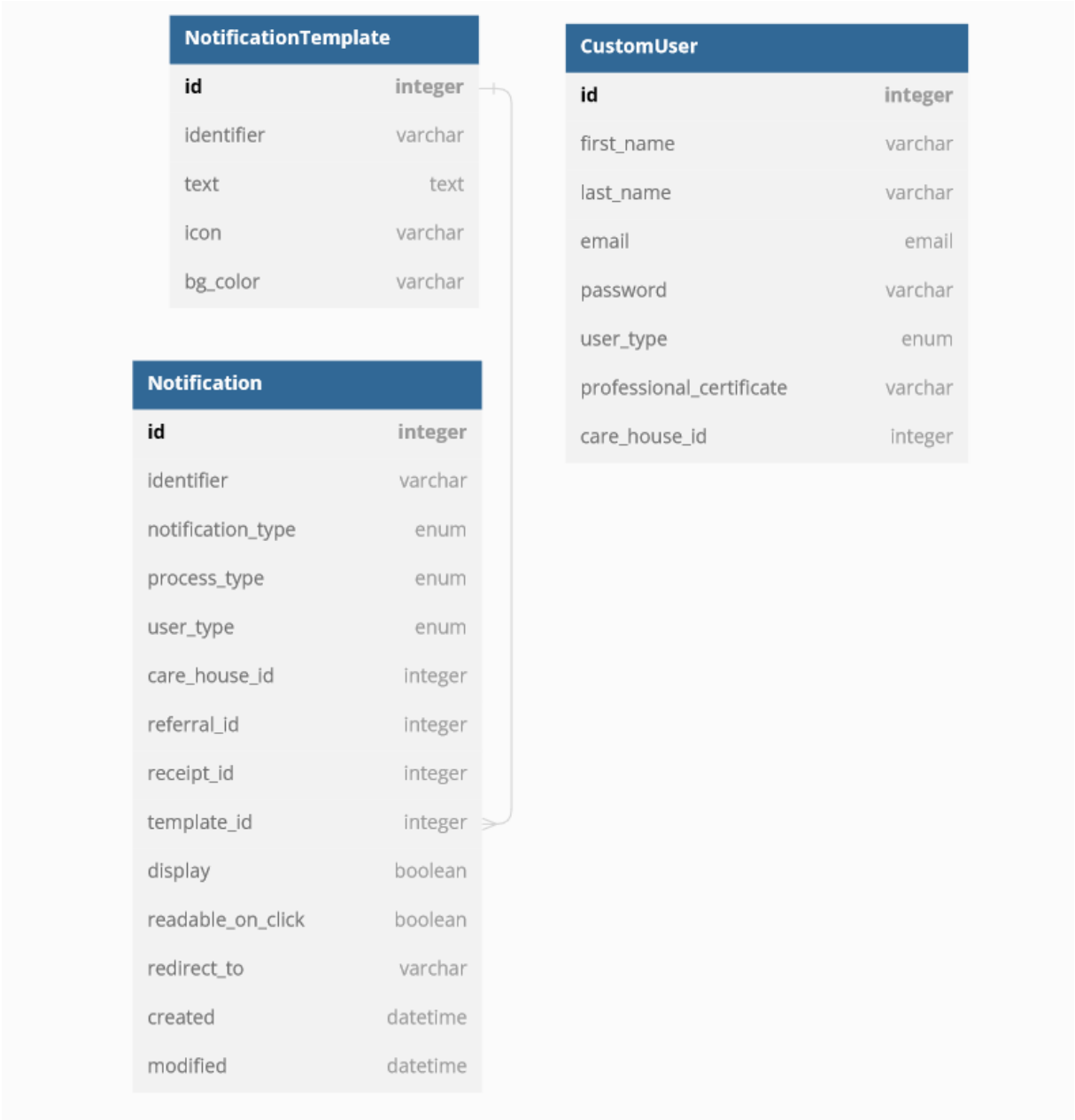

Figure 4.1: Database Model - User Management Section

Figure 4.1 contains three tables: CustomUser, NotificationTemplate and Notification. The detailed explanation of each table can be seen below.

- **CustomUser** - Table used to identify a user in the database. The table contains the following fields: id (primary key; Auto-Increment field), first_name, last_name, email, password, user_type (enum field with the following options: Doctor, Reviewer, Care House Staff, Financial Staff or Superuser), professional_certificate (only used in case of Doctor user type), care_house_id (foreign key; only used in case of Care House Staff user type).

- **NotificationTemplate** - Table used to store data relative to notification templates that might be used in the application. Contains the following fields: id (primary key; Auto-Increment field), identifier, text, icon and bg_color

- **Notification** - Table used to store information relative to notifications sent to users. A notification is shared by user type, and in the case of Care House Staff, the notification is shared in the same Care House. Contains the following fields: id (primary key; Auto-Increment field), identifier, notification_type (enum field with the following options: "*Criação de Referenciação*", "*Indicação de Vaga*", "*Avaliação de Referenciação*", "*Criação de Recibo*" and "*Avaliação de Recibo*"), process_type (enum field with the following options: "*Referenciação*" or "*Recibo*"), user_type (enum field with following options "Doctor", "Reviewer", "Care House Staff" or "Financial Staff"), care_house_id (foreign key; only used in case of Care House Staff user type), referral_id (foreign key; used for "*Referenciação*" process type), receipt_id (foreign key; used for "*Recibo*" process type), template_id (foreign key), display (Boolean field indicating if the notification is to be displayed or not), readable_on_click (Boolean field used to indicate if the notification is to be marked and read when clicked in the application), redirect_to (application URL to where the user will be redirected once the notification is clicked), created and modified.

4.1.2 Entities

In this model section, the tables are related to the different types of institutions that will be available for interaction in the application. These institutions can be care houses or medical institutions.

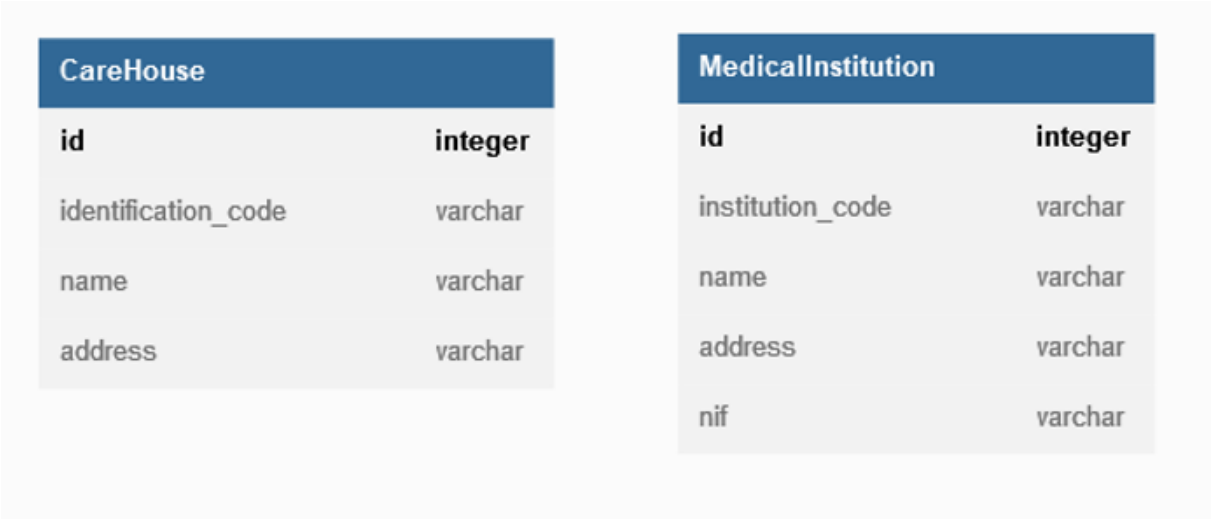

Figure 4.2: Database Model - Entities Section

Figure 4.2 represents the model section for the Entities and contains two tables, CareHouse and MedicalIntitution. The detailed information of each of these tables can be found below:

- **CareHouse** - Table used to represent a care house in the database. Currently there are three care houses that the HFF works with, namely *Casa de Saúde da Idanha*, *Casa de Saúde Santa Rosa Lima* and *Casa de Saúde do Telhal*. This model facilitates the possibility to add new care houses to the application. The table contains the following fields: id (primary key, Auto-Increment field), identification_code, name and address.
- **MedicalInstitution** - Table used to identify medical institutions in the application. These institutions are, for example, hospitals that send their mental health patients to be taken care by the HFF, which then sends them to one of the care houses for further treatment. Having a table containing this information facilitates the ability to add new institutions to the application. The table contains the following fields: id (primary key; Auto-Increment field), institution_code, name, address and nif.

4.1.3 Internment Management

This model section serves as a representation of the information needed to manage the internment process of the patients in the care houses. In this section, all the necessary information regarding the patient is stored in order to better manage the patient process from the moment of the referral until the eventual leave or discharge of treatment.

Figure 4.3: Database Model - Internment Management Section

Figure 4.3 contains the tables created for this section and respective relations and fields. This section is composed of 5 tables: Referral, Patient, CareHouseInternment, InternemnetStatus and ActivityLog. The detailed information about each of these tables can be seen below:

- **Referral** - Table used to identify a referral process. A new entry is added to this table when a referral is created by a Doctor or a Superuser. The status of a referral changes according to the actions performed by the Reviewer and the Care House Staff. The referral contains the following fields: id (primary key; Auto-Increment-Field), identifier, patient_id (foreign key), origin_institution_id (foreign key), care_house_id (foreign key), typology (enum value with the following options: "I", "II", "III"), admission_diagnosis, disease_type (enum value with the following options: "*1 - Rabilitação*", "*2 - Alcoologia*", "*3 - Agudos*", "*4 - Psicogeriatria*" and "*5 - Psiquiatria Longa Duração*"), process_number, responsibility_term (file), supersivion_scale (file), family_situation, relative_name, relative_kinship,

relative_contact, social_assistant, supervision_grade (enum field with the following options: "*Sem/Pouca Supervisão*", "*Bastante/Muita Supervisão*" and "*Alta Supervisão*"), referral_motive, other_diagnosis, social_situation, internment_duration (enum value with the following options: "*Curta Duração*", "*Média Duração*" and "*Longa Duração*"), medication, referral_date, approval _date, current_status (foreign key), referred_by (foreign key), doctor_name (auto filled if the referral is created by a Doctor user type) and doctor_profession_certificate (auto filled if the referral is created by a Doctor user type);

- **Patient** - Used to store the individual information regarding a patient. This information is stored in order to be used in case the patient returns to a care house with a new internment process. This table has some fields that are copied from the Hosix system in order to maintain data consistency. The table contains the following fields: id (primary key; Auto-Increment field), name, sns_number, social_security_number, phone_number, gender (enum field with the following options: "*Masculino*", "*Feminino*" and "*Indeterminado*"), address, postal_code, locality, country (enum field where the options are the list of countries using the ISO 3166-1 alpha-2 encoding [11]), nationality (enum filed with the same options as the country field), birth_date, subsystem (enum field where the options are the codes for every health care subsystem in Portugal), created and modified;

- **CareHouseInternment** - Table used to represent a Care House Internement process in the database. The internment process is initiated once the referral process is concluded with an approval evaluation. The patient is then transferred to the care house indicated in the referral process. The table contains the following fields: id (primary key; Auto-Increment filed), identifier, referral_id (foreign key), admission_date, admitted_by_id (foreign key), current_status_id (foreign key) and leave_date;

- **InternmentStatus** - Table used to identify a status and its possible transitions both for the Referral and the Internment process. The table contains the following fields: id (primary key, Auto-Increment field), label, name, type (enum field with the following options: "Referral" and "Internment"; used to identify to which process the status corresponds), next_states (list field with the labels of the next possible states) ad prev_states (list field with the labels of the previous states);

- **ActivityLog** - Table used to identify the activities partaken by the patient while in the care of a care house. These activities can be external consultations on another hospital, returns to the care house and permanent exists from the care

house. The table contains the following fields: id (primary key; Auto-Increment field), internment_id (foreign key), executed_by_id (foreign key), activity_type (enum field with the following options: "*Entrada*", "*Saída Temporária*", "*Retorno*" and "*Saída*"), action, description and log_date.

4.1.4 Financial

This section contains the tables regarding the financial components of the active internments in the care houses. This part of the model stores data like the invoice information for every month in each care house, as well as the information regarding the amount of money each institution has to pay for the care of their patients.

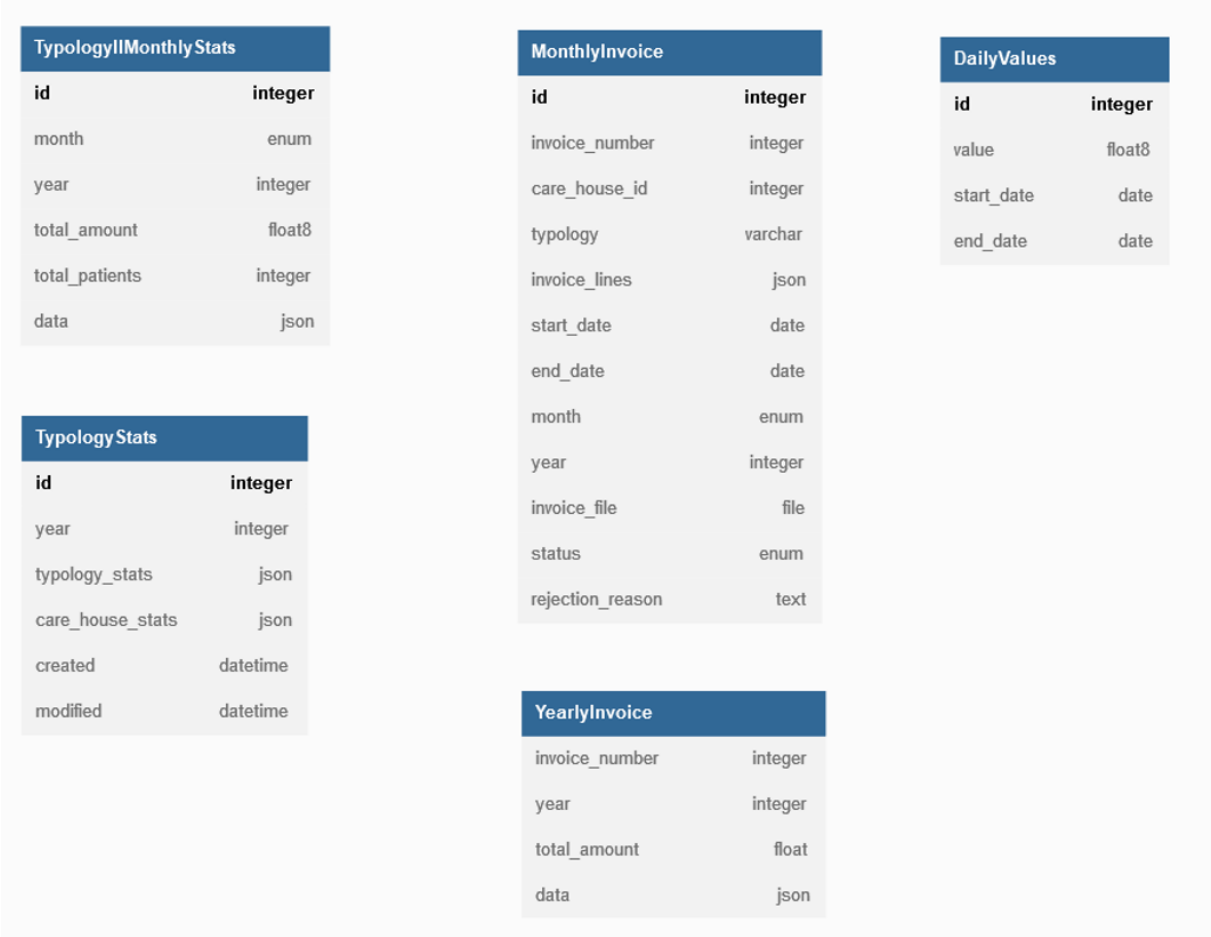

Figure 4.4: Database Model - Financial Section

The Figure 4.4 contains five tables, each one corresponds to a different type of financial data that needs to be stored for access in the application. The tables indicated are: MonthlyInvoice, TypologyStats, TypologyIIMonthlyStats, YearlyInvoice and DailyValues. The detailed description of each table can be found below:

- **MonthlyInvoice** - Stores the monthly patient data for each typology in the respective care house. Once the invoice is created by the Care House Staff, the Financial Staff must evaluate the invoice. If the invoice is invalid, then the Care House staff must proceed to the creation of a new invoice. The table contains the following fields: id (primary key; Auto-Increment field), invoice_number, care_house_id (foreign key), typology, invoice_lines (a JSON field that stores the individual data of each patient in the invoice), start_date, end_date, month, year, invoice_file (a file provided by the Care House Staff with the proper invoice), status (enum field with the following options: "*Dados Temporários*", "*Aguarda Pagamento*", "*Pago*" and "*Reprovado*") and rejection_reason (only used in case of an invoice rejection);
- **YearlyInvoice** - Stores the yearly values for each patient in every care house. The values are organized by month and care house. The table contains the following fields: id (primary key; Auto-Increment field), invoice_number, year, total_amount and data (a JSON field with the invoice lines)
- **DailyValues** - Table used to store the values to be applied each day for an interned patient. The values are applied while in the date range stored. The table contains the following fields: id (primary key; Auto-Increment field), value, start_data and end_date;
- **TypologyStats** - Table used to store stats regarding the interned patients. Each entry stores data relative to typology stats and care house stats. The typology stats are divided by each typology while the care house stats are divided per care house. The table contains the following fields: id (primary key; Auto-Increment field), year, typology_stats (JSON data containing the typology stats data), care_house_stats (JSON field containing the care house patient's data) created and modified;
- **TypologyIIMonthlyStats** - Table used to store the information regarding the money that each institution has to pay monthly to the HFF for the care of their patients. The table contains the following fields: id (primary key; Auto-Increment field), month, year, total_amount, total_patients and data (JSON data with the detailed amount of patients and money owed per institution);

4.2 Django Project

After defining the database model to be used by the application to store important data, the next step of development consists of the creation of the Django Project that

will serve as the code base.

A Django Project is built using the Python programming language and consists of a set of applications that represent the different section of the project. These applications will then be represented in the Django Admin page, organizing the defined Database Model in an easy to understand representation.

This section contains all the necessary information regarding the project that is necessary in order to understand how the project was developed and to facilitate the understanding of the web application mechanics. The subsections below contains the description of the project's file structure, a short description and examples from the Django Admin Panel, a description of the Web Application and its functionalities, how the application was tested and finally the deployment description.

4.2.1 File Structure

When a Django Project is created, a set of files are created with the project configurations. These files are created inside a Python package.

After the project creation, the Django Applications can be created using the command line. These applications are created in the form of Python packages with a set of files that serve as the definition of models, views and controllers. The applications must be registered in the Django settings file in order to be used by the project. This is a manual process.

Besides the applications, a Django Project also supports the creation of Python packages that can be used in the remaining packages. These packages must be created manually and do not need to be declared in the settings file.

When the Django Project and respective applications are created, it is then necessary to create a requirements file which will contain the project dependencies. The file creation can be done via the command line using a Python command.

The main file structure will then be composed of the configurations package, the applications and the requirements file.

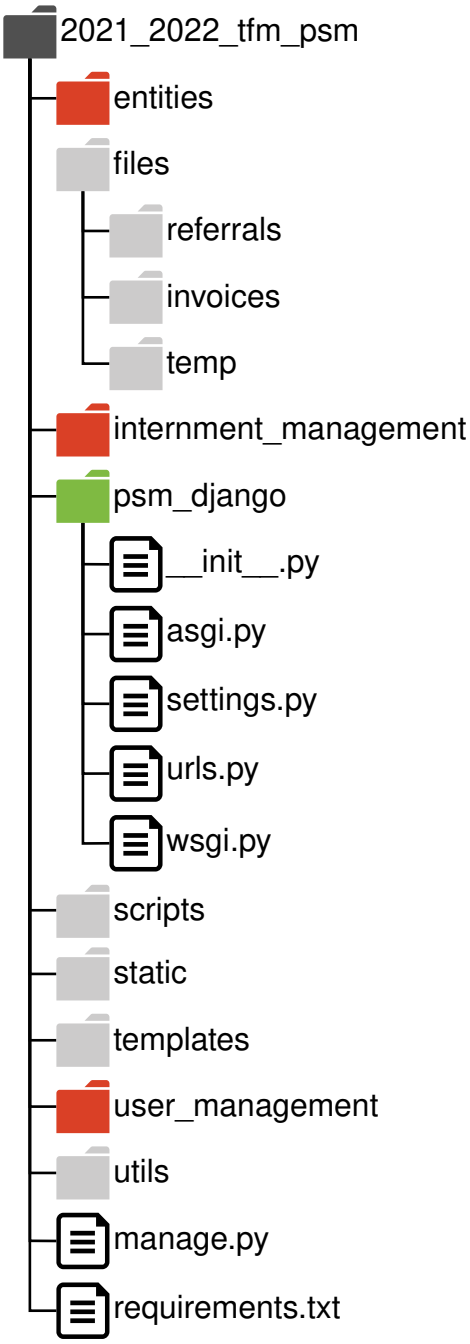

Figure 4.5: Main File Structure

Figure 4.5 is a representation of the main file structure present in the project. The project consists of four applications (entities, financial, internment_management and user _management), a configuration package (psm_django) and a few extra folders and packages (files, scripts, static, templates and utils).

4.2.1.1 Configuration Package

The Configuration Packages of a Django project consists of a set of files that are necessary in order to define the settings of the project. This package is created alongside the project.

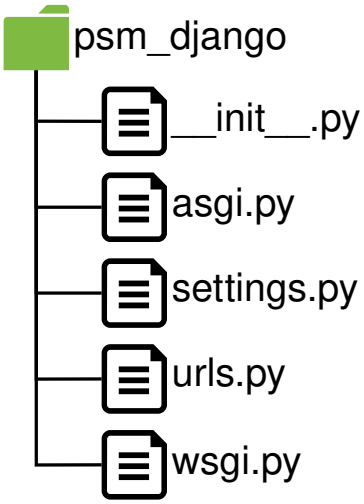

Figure 4.6: Configuration Package File Structure

Figure 4.6 is a representation of the configuration package and the files it contains. The package contains the following files.

- asgi.py - Contains the necessary project configuration to run the project in ASGI mode.
- settings.py - The main project configuration file. Contains information like the database connection, Django Applications to use, allowed hosts that can access the project once deployed, among other information.
- urls.py - File containing the available paths to access the pages and APIs of the application.
- wsgi.py - Contains the necessary project configuration to run the project in WSGI mode.

4.2.1.2 Applications

The Django Applications consist of a set of folders and files that contain the declarations of Models, Views and Controllers of the project. A project can be made of

multiple applications, as they serve as a way of creating different sections in order to facilitate the organization of the project. These applications will then be represented in the Django Admin Panel as a way of interacting with the data stored in the database.

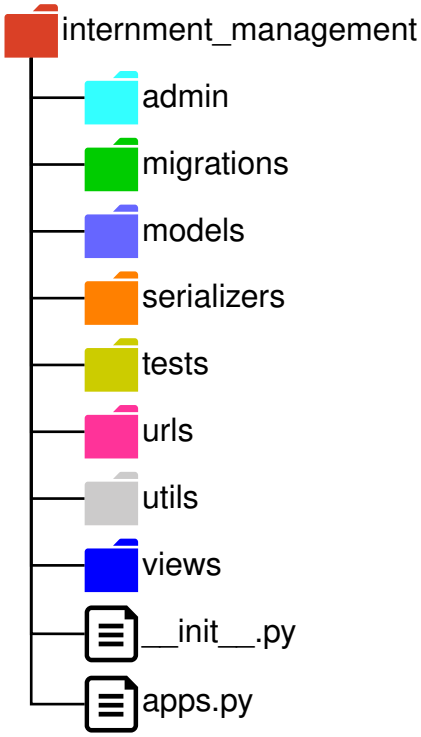

Figure 4.7: Applications File Structure

Figure 4.7 is a representation of an example of a complete Django Application file structure. The application contains the following files and folders:

- admin - Contains the files with the configuration of the model representation in the Admin Panel for every model in the application. These configurations include: list of fields to display in the list view, custom functions to execute over the data, filters for the list view, among other settings.
- migrations - This folder contains all the migrations necessary to create, alter or remove fields and tables from the database. Since Django uses a code-first model, this folder is the representation of all the modification to the database regarding the models in the respective application. These migrations are created using an auxiliary command line tool, i.e. *python manage.py makemigrations*.
- models - Folder containing the models for the application, which will be converted to database tables once the migrations are created. In each model file is

declared the fields and respective types for each table. A model file also accepts certain configuration options like the table name in the database, the name to display in the Admin Panel, among other options.

- serializers - This folder contains a set of files that declare classes that are used for validating the bodies or query strings for incoming request to the application. These classes are only used for developer defined paths and are not used in the Admin Panel.
- tests - The tests folder contains the declaration for all the tests that need to be done in order to test the application.
- urls - Defines the paths for the pages and APIs present in the respective application.
- views - Represents the Controllers in the MVC (Model View Controller) model. Contains all the controllers for the respective application.

4.2.1.3 Extras

Alongside the Django Applications and configuration package, a Django project also supports the creation of extra folders and files that can be used by the applications. These folders do not need to be declared in the settings (except for template folders) file and to use functions written in Python files all that needs to be done is import them. For this project, the extra folders created were the following:

- files - Used to store files that are used in the Web Application. In this project there is a need to store some files related to the referrals and invoices. These files are uploaded to the back-end via the web application and stored in this folder. In the models mentioned before, some of the fields are file fields, which point to the path of a file stored in this folder.
- scripts - Folder containing custom scripts that can be run in order to execute certain actions that cannot be performed in the Admin Panel due to its limitations. In order to be able to run these scripts it is necessary to install a Python library called "django-extensions"[12].
- static - Folder containing the static files that will be served to the client. This folder includes files with extensions like HTML, CSS and JavaScript. This folder is declared in the settings file and it is created and updated with the latest files via a command in the command line.

- templates - Templates folder that contains the views that will be showed in the client side. In order to create the views there are 2 possible ways: option one is to create a single templates folder (like in this project) and organize the files inside in order to be used by the applications (this folder needs to be indicated in the settings file in the templates variable), option two is to create a templates folder inside every Django Application and use the files from there. Using option one, all the files are in a single place which makes it easier to access and visualize. The HTML files created in this folder can then be used by the controllers to render the page on the server side and send them to the client.
- utils - A folder containing auxiliary functions that can be used in the applications.

4.2.2 Development Features

The Django Framework provides a set of tools, such as the Django ORM (Object-Relational Mapping) and the Django Templates, that aid developers in the creation of their projects. These tools provide the ability to interact with elements such as the database or the HTML templates, without the need to use multiple frameworks.

4.2.2.1 Django ORM

The Django ORM is a feature included in the Django Framework that allows the creation of Models that will then be mapped to tables in the project's database. This tool allows the execution of operations such as search, creation, update or deletion of records in the database, abstracting the developer from using raw SQL.

```
1  class CareHouseInternment(models.Model): # The identifier of the internment
       .
2      identifier = models.CharField(max_length=64, unique=True) # The
       referral of
3      the internment. referral = models.ForeignKey(Referral,
4      on_delete=models.CASCADE) # The date of admission. admission_date =
5      models.DateTimeField(null=True, blank=True) # The user who admitted to
       the
6      care house. admitted_by = models.ForeignKey(CustomUser,
7      on_delete=models.CASCADE) # The current status. current_status =
8      models.ForeignKey(InternmentStatus, on_delete=models.CASCADE) # The
       date of
9      the end of the internment. leave_date = models.DateTimeField(null=True,
10     blank=True)
```

Listing 4.1: Model Example

The example shown in Listing 4.1 is a representation of a Python Class that is mapped to a table in the project's database. The class fields represent the columns of the table and values represent the data type (e.g. CharField represents VARCHAR) and constraints (e.g. max_length represents the maximum number of characters present in a VARCHAR).

After defining the models, all the operations that interact with the database are executed on instances of the defined class.

```
# Record Creation (Option A) care_house = CareHouse.objects.create(
identification_code=1, name="Casa de Saúde da Idanha", address="Rua A" )

# Record Creation (Option B) care_house = CareHouse( identification_code=1,
name="Casa de Saúde da Idanha", address="Rua A" ) care_house.save()

# Record Select (Single) care_house =
CareHouse.objects.get(identification_code=1)

# Record Select (Multiple) care_houses = CareHouse.objects.filter(name="
    Casa de
Saúde da Idanha")

# Record Select (All) care_houses = CareHouse.objects.all()

# Record Update (Option A) care_house.name = "Casa de Saúde da Idanha"
care_house.save()

# Record Update (Option B) CareHouse.objects.all().update(address="Rua B")

# Delete Record CareHouse.objects.get(identification_code=1).delete()
```

Listing 4.2: Operations Example

The examples shown in Listing 4.2 are a representation of the operations that can be executed on the table records. Some of the operations have different variations on how they can be executed, while having the same result.

These operations can then be executed in the views (Controllers in the MVC) that can render the templates (Views in the MVC).

```
@sync_to_async() def receipt_history_page(request): # Verify if the method
    is
allowed. if request.method == 'GET': # Obtain the user that made the
    request.
        user = request.user  # type: CustomUser # Verify if the user is
```

```
4         authenticated. if not user.is_authenticated() return redirect("/
      login/")
5
6         # The active status. active_status = ['paid', 'rejected'] # Obtain
      the
7         invoices data. invoices = [ { 'id': i.invoice_number, 'typology':
8         i.typology, 'start_date': i.start_date.strftime('%d/%m/%Y'),
9                 'end_date': i.end_date.strftime('%d/%m/%Y'),
10                'total_patients': i.invoice_lines['total_patients'],
11                'total_amount': i.invoice_lines['total_amount'], 'status':
12                i.get_status_display() } for i in MonthlyInvoice.objects.
      filter(
13                care_house=user.care_house, status__in=active_status ) ] #
14                Define the context to be sent to the page. context = {
15                'invoices': invoices, }
16
17        return render(request, 'finances/html/care_house_receipt_history.
      html',
18        context=context)
19
20    else: return redirect('/login/')
```

Listing 4.3: View Example

The example shown in Listing 4.3 is a representation of a view that renders a templates. The `@sync_to_async` serves to indicate that this view is to be executed asynchronously. The controller receives a request parameter that contains information such as the HTTP method called, the user that made the request, and other data. The user can also be validated in the controller. In this case, if the user is not authenticated, they are redirected to the Login page.

The data obtained in the view is sent as context to the templates. This context represents data that can be access by the Django Templates in order to render dynamic pages.

```
1  {% extends 'base/html/base_index.html' %}
2
3  {% block content %}
4      <div class="flex-row d-flex justify-content-between mb-3
5      align-items-center"> </div> <!-- Basic Bootstrap Table --> <div
6      class="card"> <div class="card-header d-flex justify-content-between
7      align-items-baseline"> <div class="h5">Histórico de Recibos</div> </div
       >
8      <div class="card-body"> <table id="receipts_list" class="table"> <thead
       >
```

```
<tr> <th>Tipologia</th> <th>Inicio de Contabilização</th> <th>Fim de
Contabilização</th> <th>Total de Pacientes</th> <th>Valor total</th>
<th>Estado</th> <th>Detalhes</th> </tr> </thead> <tbody id="">
{% for i in invoices %}
                    <tr> <td>{{ i.typology }}</td> <td>{{ i.start_date
                        }}</td> <td>{{ i.end_date }}</td> <td>{{
                        i.total_patients }}</td> <td>{{ i.total_amount
                        }}</td> <td>{{ i.status }}</td> <td><a
                        href="javascript:void(0);" data-invoice-id="{{
i.id
                        }}">Ver Detalhes</a></td> </tr> {% endfor %}
            </tbody> </table> </div> </div> <!-- Basic Bootstrap Table
 -->
        {% endblock %}
```

Listing 4.4: Template Example

Listing 4.4 represents a templated rendered using Django Templates. This allows the creation of a dynamic page that changes according to the data that is sent to the template. The Django Templates tool also allows the use of certain functionalities, such as extending another template in order to reduce the use of repeated code.

The `{%extends ...%}` directive indicates which template is being extended and the `{% block ...%}` directive indicates in which section of the template the new data will be inserted. Besides these directives, Django Templates also allow the use of operations like `for` and `if` over the data sent from the controller, using a syntax similar to the Mustache[13].

4.2.3 Django Admin

The Django Admin panel comes integrated with a Django Project and allows users with certain permissions to manipulate the data stored in the project database with ease. A user must be granted access to this panel by a user who already has access to it.

This panel serves as a visual representation of the project database and allows actions like view, create and update records in the database abstracting the user from queries. The tables and data visualized in this panel must be defined in the admin folders of the Django Applications and the application must be declared in the settings file.

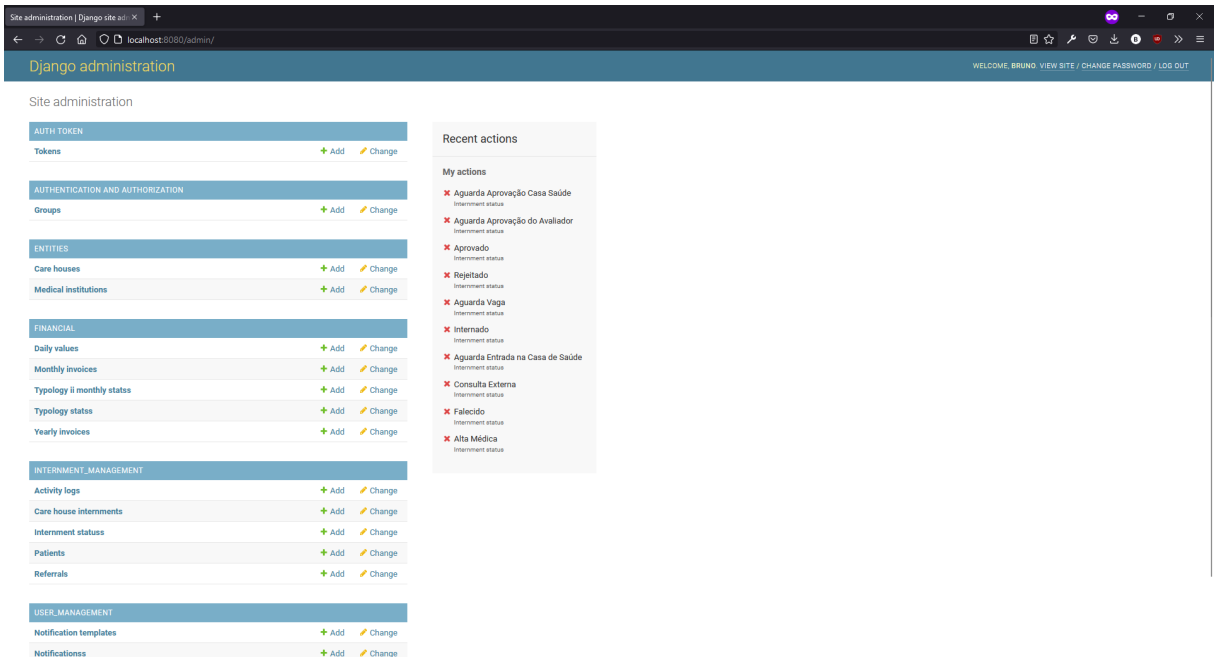

Figure 4.8: Admin Panel Dashboard

Once the user is logged in, they are redirected to the dashboard shown in Figure 4.8. This dashboard contains the list of applications in the project along with the respective models (tables) and a list of recent actions made by the user.

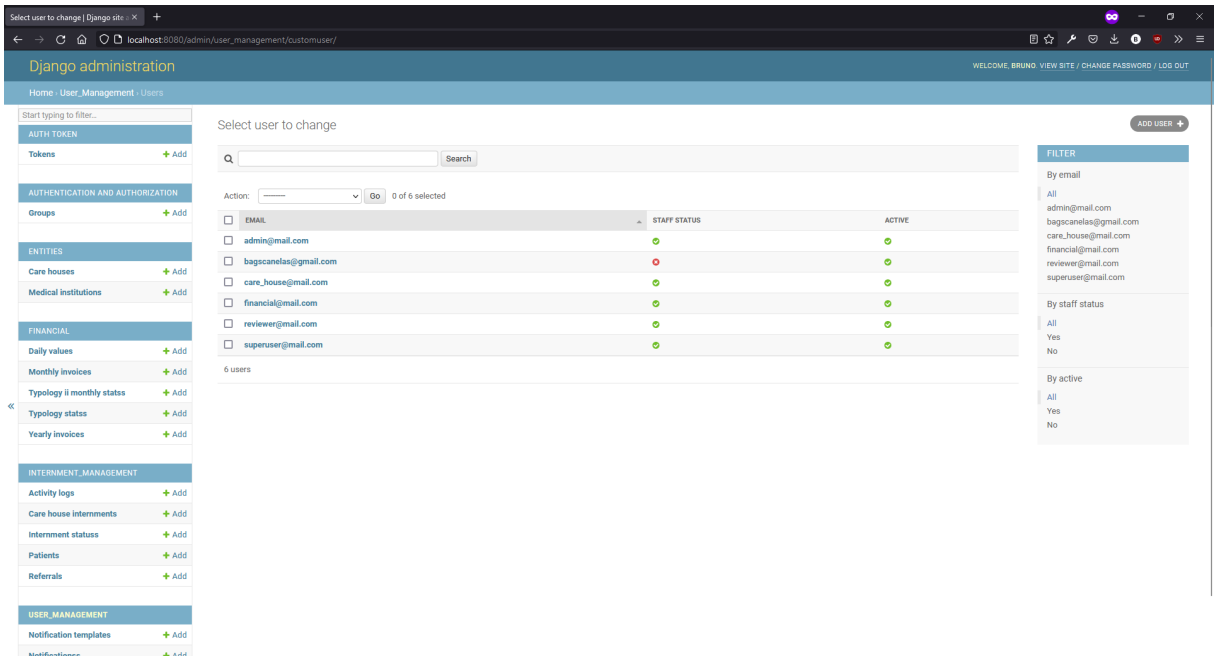

Figure 4.9: Admin Panel Record List

Clicking one of the models will take the user to a page with the list of records in that table, like in Figure 4.9. The user can use the filter in the right side panel to filter the

list of records and also the search bar above for specific searches. In this page, the user can also add a new record to the table.

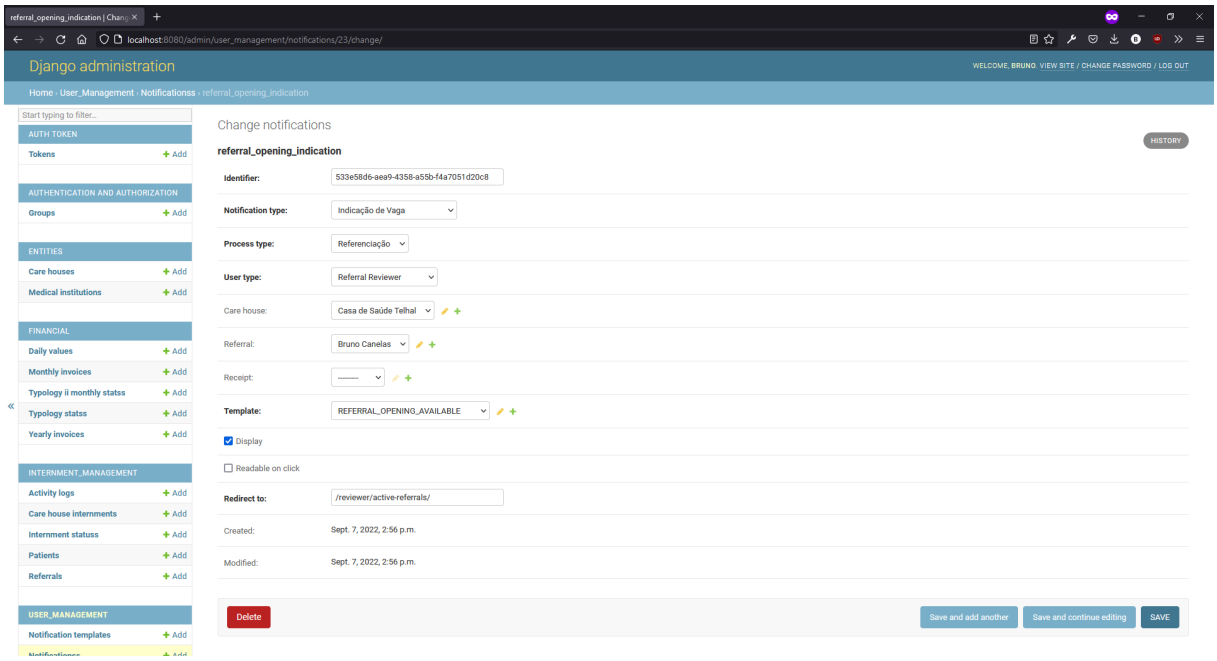

Figure 4.10: Admin Panel Record Details

By clicking one of the records, the user will be shown all the fields of the record along with the respective values for that specific record, as shown in Figure 4.10. In this page, the user can manipulate the data in the record or delete it. The user can also check the history of actions made to the specified record.

4.2.4 Web Application

The project's web application was created using the Django Templates system. This system is included in a Django Project and allows the creation of server-side[14] rendering web pages using a Mustache like syntax. All the pages were created in the templates folder, alongside with all the necessary JavaScript and CSS files. The web pages were created using the mock-ups shown in chapter 2 of this document.

This section will describe all the pages created and all the differences between them and the mock-ups indicated.

4.2.4.1 Institution Psychiatrist

The Institution Psychiatrist is represented by the Doctor user type in the application. For this user type, it was necessary to create only two pages in the application, given

the fact that this user type can only create referrals and verify the referral history.

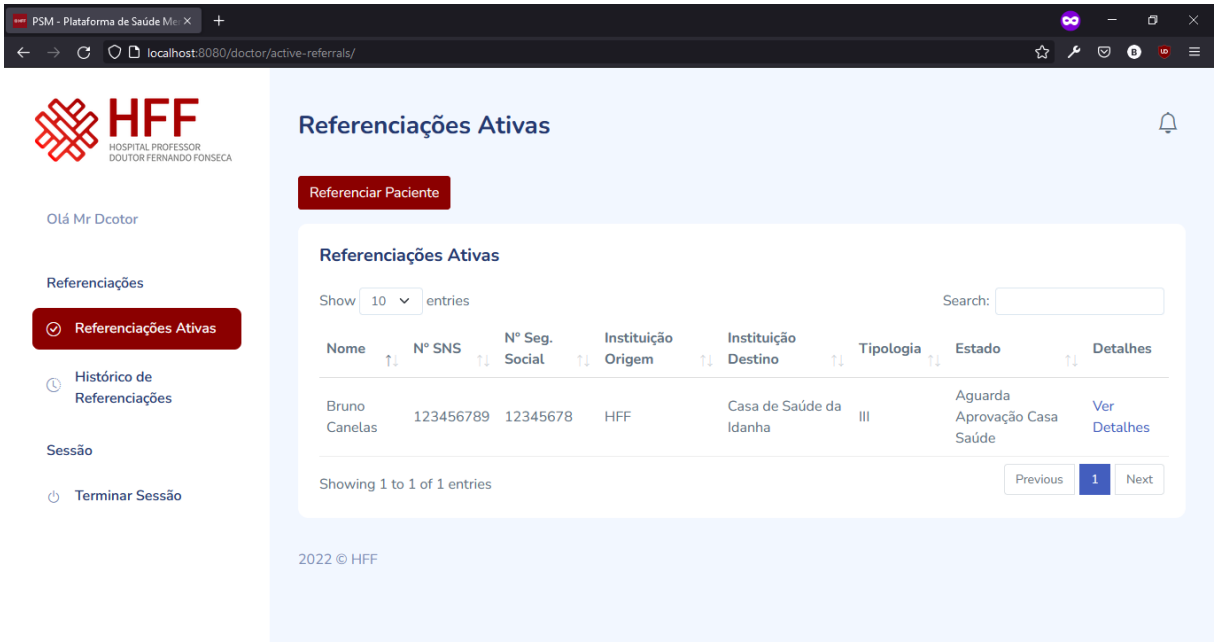

Figure 4.11: Doctor - Active Referrals

Figure 4.11 is the Active Referrals page for the Doctor user type. In this page the user is able to verify the list of active referrals, along with the ability to create a new referral. The referrals list only shows referrals that have not yet been approved or rejected by the remaining user types. The user is able to filter the list in order to search for a specific referral, sort the data by a chosen column or verify the full information regarding the referral.

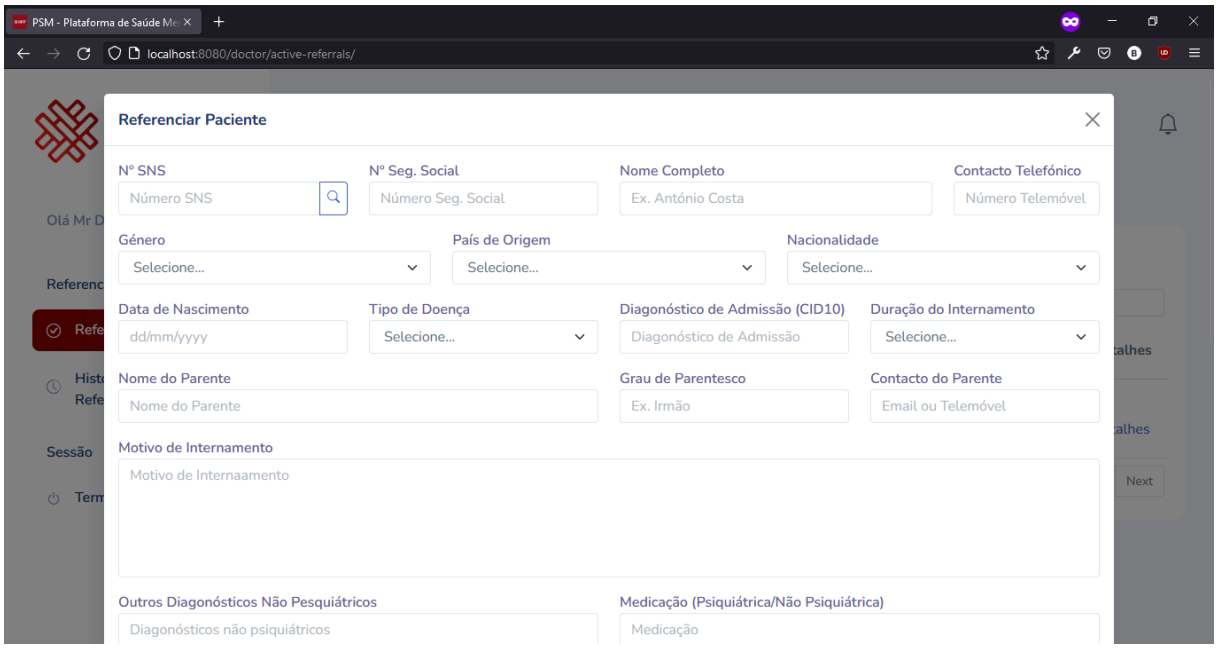

Figure 4.12: Doctor - Active Referrals

Figure 4.12 represents the form a user needs to fill in order to create a new referral. The form contains a validation created using the Bootstrap toolkit that will display what fields were incorrectly filled before submitting the form. The user is also able to search for the data of a specific patient using the SNS Number field. This action will search for the patient data in the Hosix system (production environment) or a test database (test environment). If the patient is found, then the patient data fields will be filled automatically, if the not the user is then asked if the new patient is to be inserted in the database when the form is submitted.

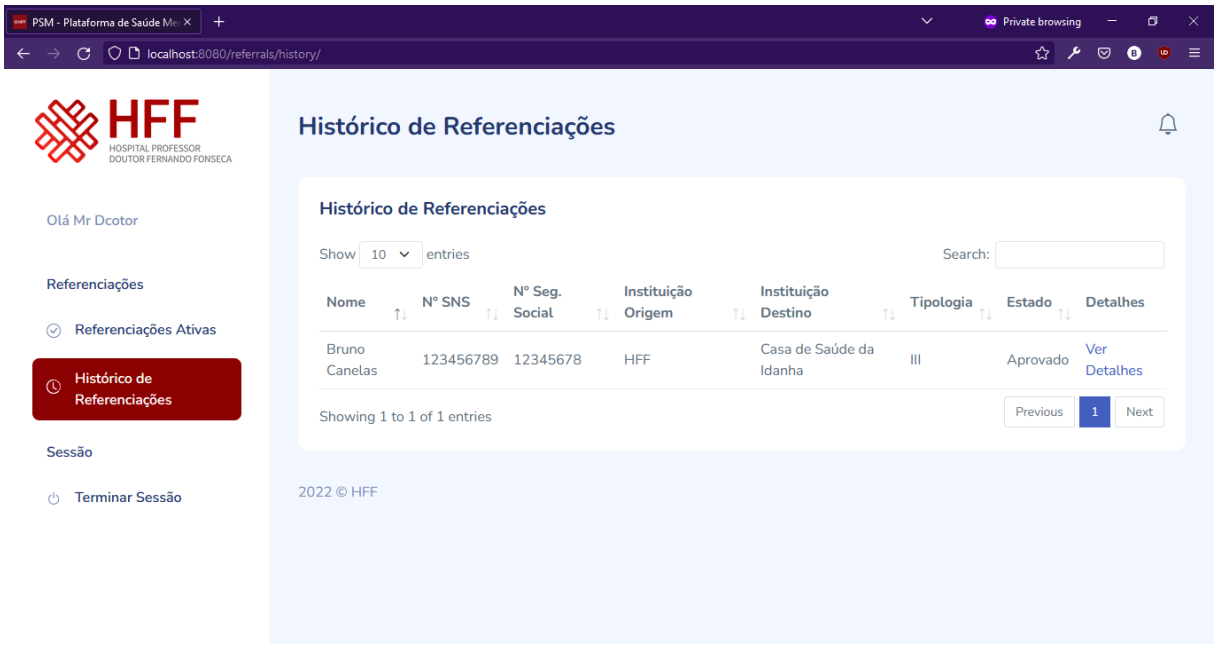

Figure 4.13: Doctor - Referral History

Figure 4.13 represents the Doctor Referrals History page. This page only contains the list of referrals that are already accepted or rejected by the remaining user types. In here the user is only able to visualize data and cannot create or update any records. The list in this page allows the user to perform the same filtering and sorting actions that are present in the Active Referrals page.

When comparing the pages created with mock-ups created, there are some differences in the organization of the data. In the mock-ups it was only indicated one page containing both the active referrals and previous referrals. Separating the two types of referrals allows the user to visualize the referral data with a better organization of the information, without the need for excessive filtering over the records in the list.

4.2.4.2 Reviewer

The Reviewer user type has the ability to evaluate referrals and indicate if they are suited for the indicated care house or not. For this, two pages were created for this user type.

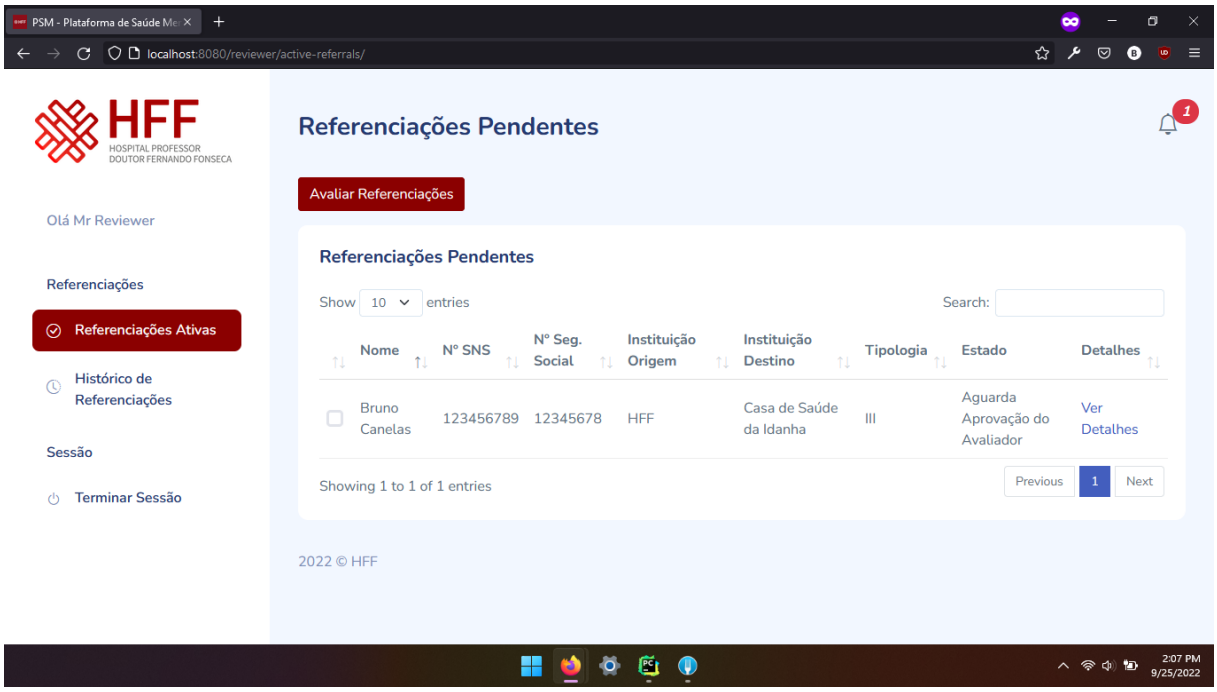

Figure 4.14: Reviewer - Active Referrals

Figure 4.14 represents the Reviewer Active Referrals page. This page contains all the referrals created by Doctors that have already been approved by the Care House. The items in this listing can be selected for a batch evaluation. In this page the user is able to verify the referral data, as well as evaluate them. The filtering and sorting options are identical to the options present in the Doctor user type referrals pages.

Figure 4.15: Reviewer - Referral History

In the page represented by Figure 4.15 the user is able to verify the referrals that have already been accepted or rejected. The user is only able to visualize data in this page and cannot modify it or add new records, similarly to the Doctor Referral History page.

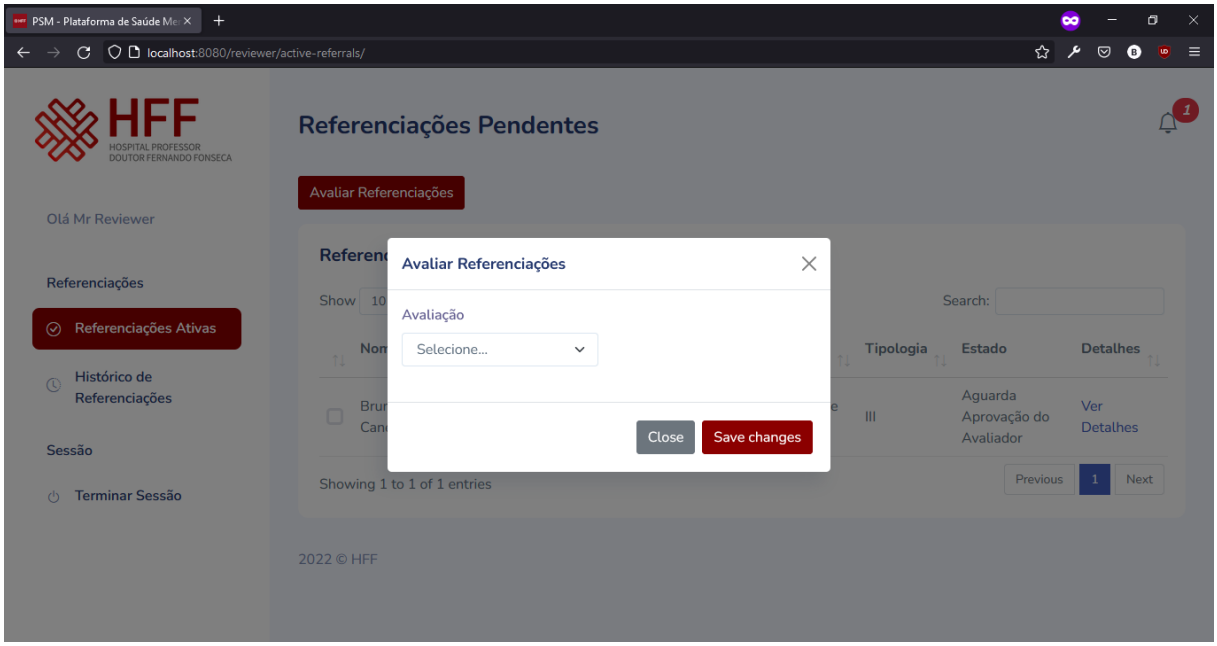

Figure 4.16: Reviewer - Referral Evaluation

Figure 4.16 represents the evaluation pop-up that is shown to the user once a batch of referrals is selected for evaluation. This pop-up prompts the user to a selection field where they will indicate if the referral is approved or rejected.

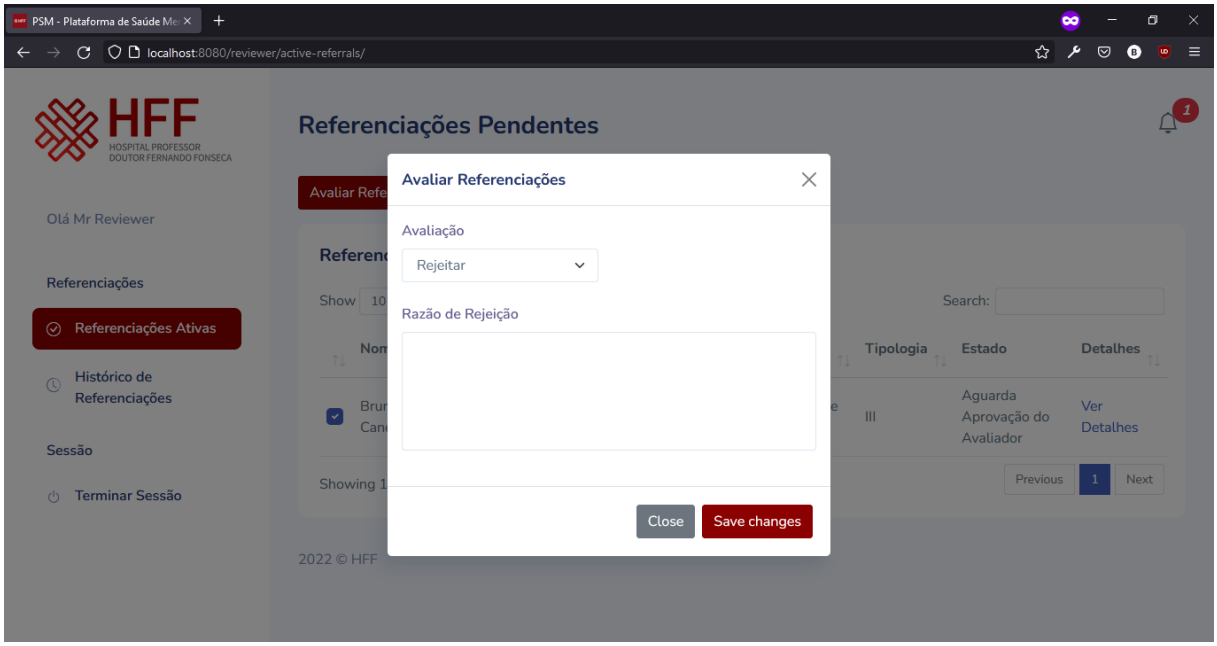

Figure 4.17: Reviewer - Referral Rejection

In page represented by Figure 4.17 the user has selected to reject the referral. In this case the user must indicate a reason to why this decision was taken. The rejection reason is only shown to the user once the option to reject the referral has been selected.

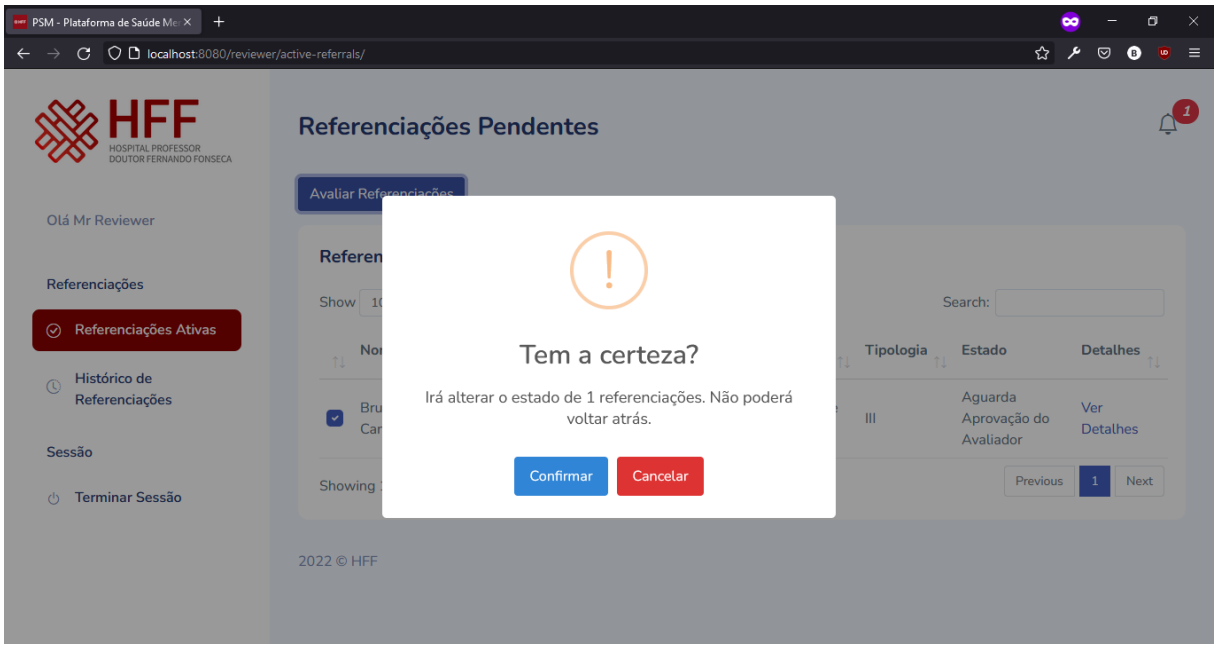

Figure 4.18: Reviewer - Evaluation Confirmation

Figure 4.18 represents the pop-up that is shown to the user once decided to submit the evaluation. The pop-up serves as a confirmation to the user action, showing the number of records that will be affected by the action.

The pages presented in this subsection are based on the mock-ups indicated in a previous chapter, the only difference being the existence of two pages that display referral listings, one for the currently active referrals and another for those that have already been approved or rejected. The decision was made for the same reasons indicated in the Doctor user type.

4.2.4.3 Care House Staff

The Care House Staff represents the users that belong to a care house and manage the patients interned in the respective care house, as well as the generation of monthly invoices with the daily care value for each patient. All the actions done by a patient must be logged in the web application in order to ensure the correct creation of invoices.

Since this type of user is the one with the most responsibilities, there was a need to create a bigger number of pages for the user to interact with. These pages are divided

into sections according to the sector they belong to. These sections are: "*Referenciações*" (Referrals), "*Gestão de Internamentos*" (Internment Management) and "*Gestão Financeira*" (Financial Management)

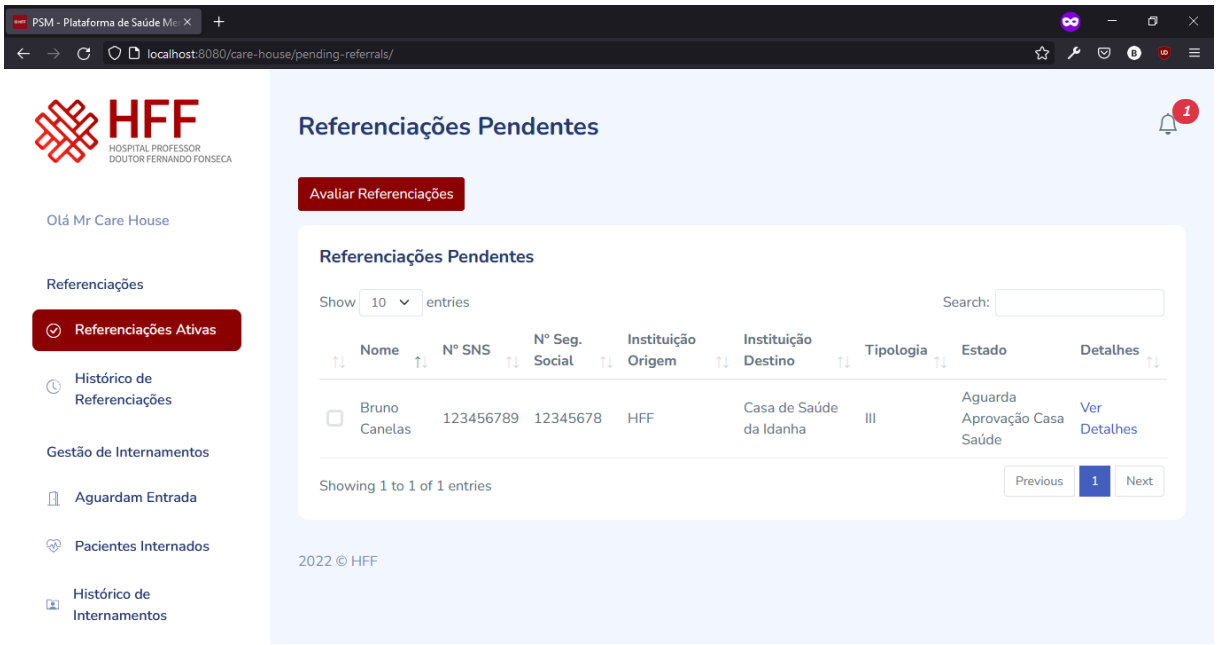

Figure 4.19: Care House - Active Referrals

Figure 4.19 represents the Active Referrals page for the Care House Staff user type. In this image the user can verify the list of referrals that are currently waiting for care house approval. The user can then select a batch of referrals and evaluate them, indicating if they are valid for internment or not.

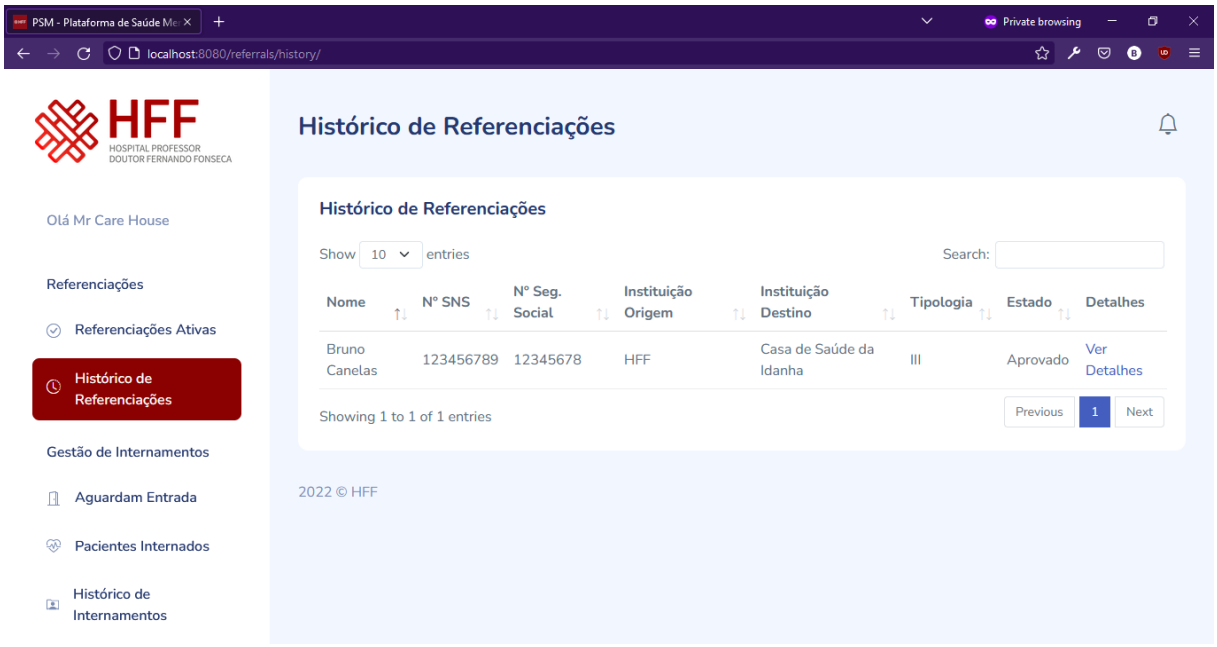

Figure 4.20: Care House - Referral History

Figure 4.20 represents the Referral History Page for the Care House Staff user type. This page allows the user to verify the referrals that have been approved or rejected, however no changes can be made to the data nor can the user create new records.

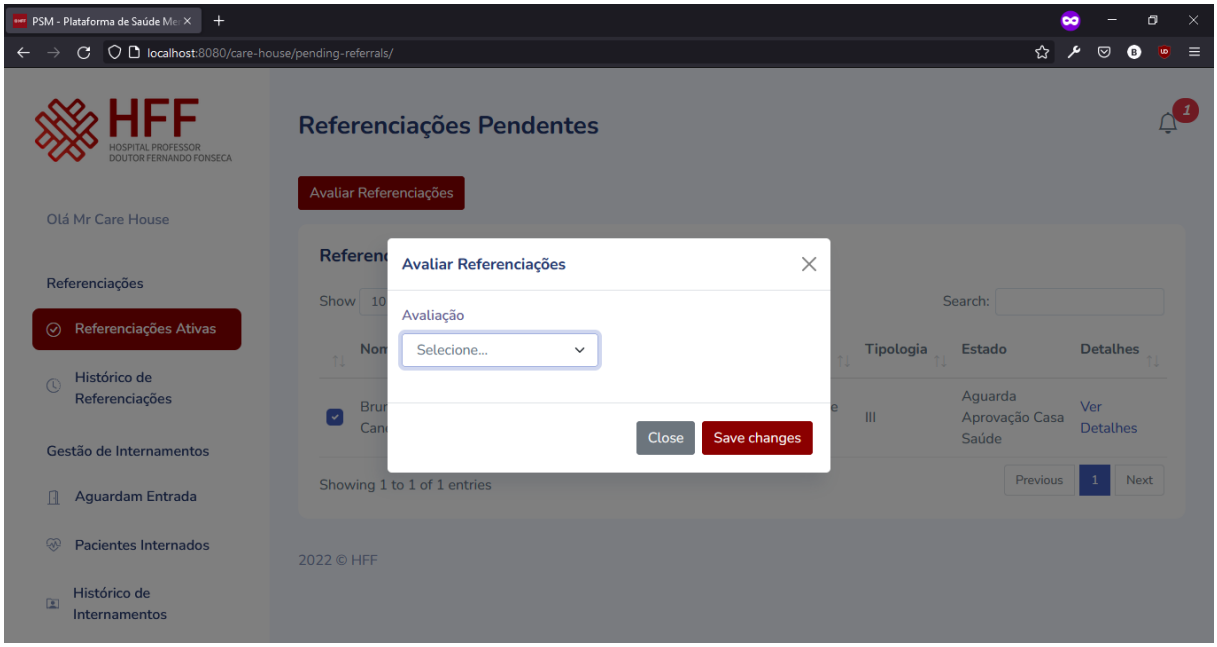

Figure 4.21: Care House - Evaluation

After selecting the batch to evaluate, the user is prompted with a pop-up where they need to indicate the evaluation result, like the one shown in Figure 4.21. The user must

then select an option from the select field.

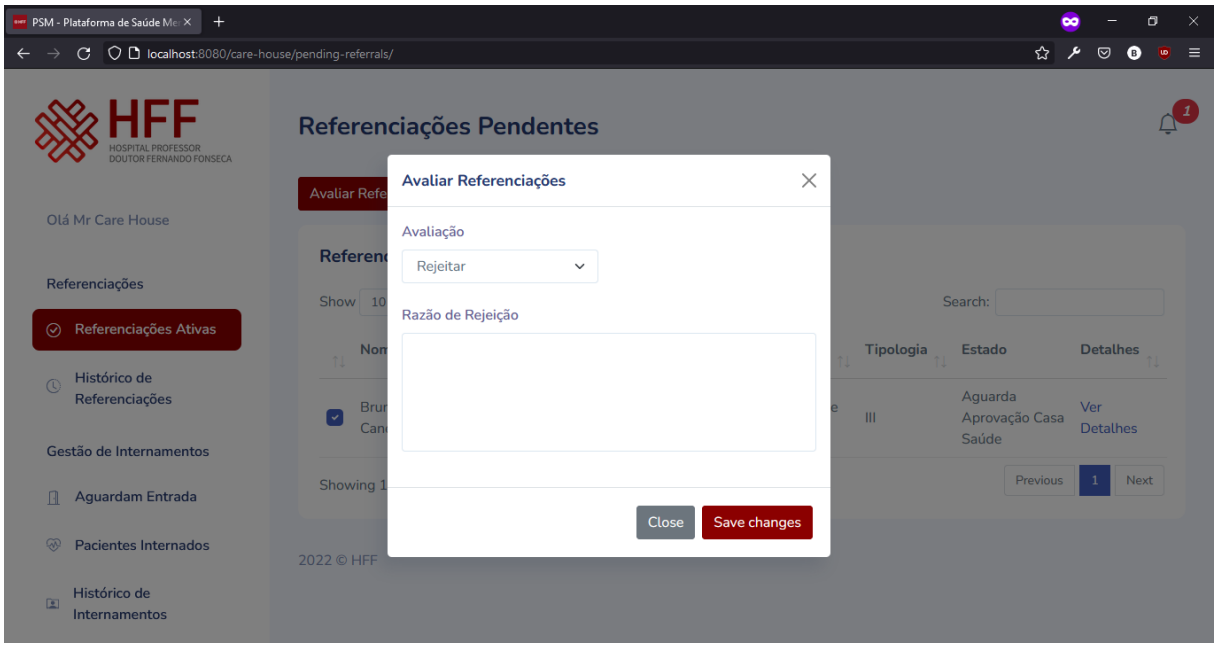

Figure 4.22: Care House - Rejection

In case the user decides to reject the referral, then the rejection reason field will appear for the user to fill, as shown in Figure 4.22.

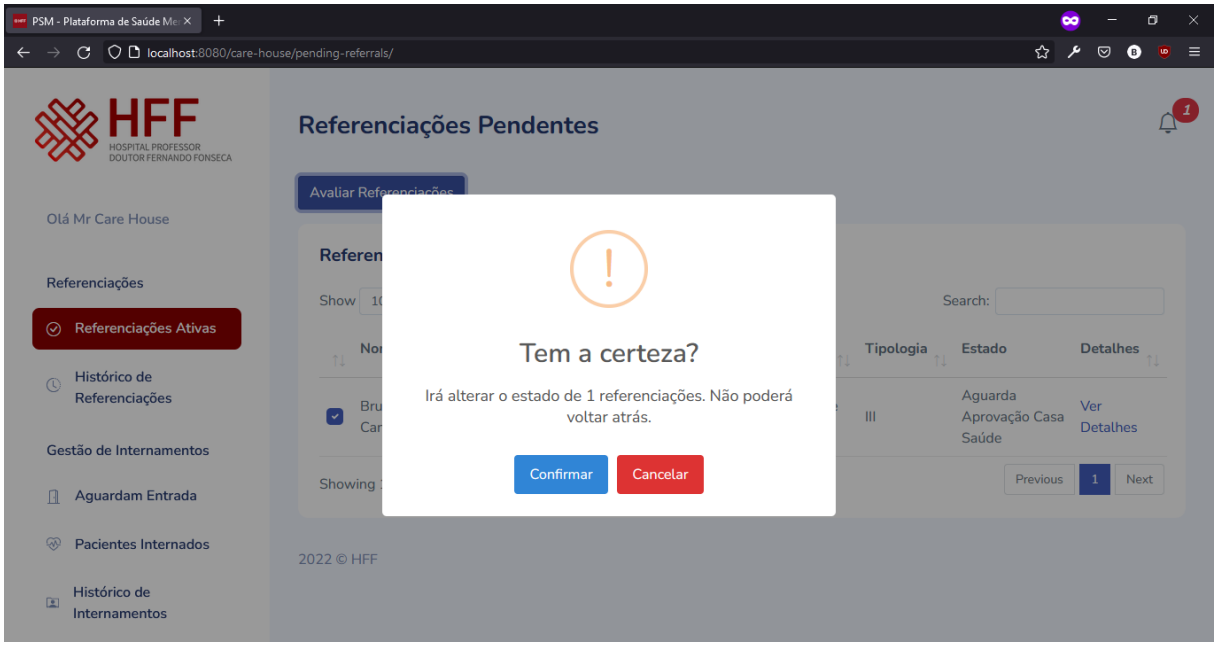

Figure 4.23: Care House - Evaluation Confirmation

Once the user has made the evaluation, the pop-up represented in Figure 4.23 will

appear as a way for the user to confirm the evaluation. The pop-up will show the number of records that will be affected by the user action.

The set of images shown in this subsection are part of the Referral section of the Care House Staff user type dashboard. The pages displayed are similar to the previous user type and just like before, the pages are similar to the mock-ups designed in a previous chapter. All the similarities and differences were indicated in the previous user type description of the pages.

Figure 4.24: Care House - Awaits Entry

Figure 4.24 is a representation of the Care House Awaits Entry. This page shows the list of patients that have been referred to a care house and approved for internment, however they have not been admitted into the care house yet. In this page the user is able to access the data belonging to a patient's referral, as well as mark the patient as interned in the care house.

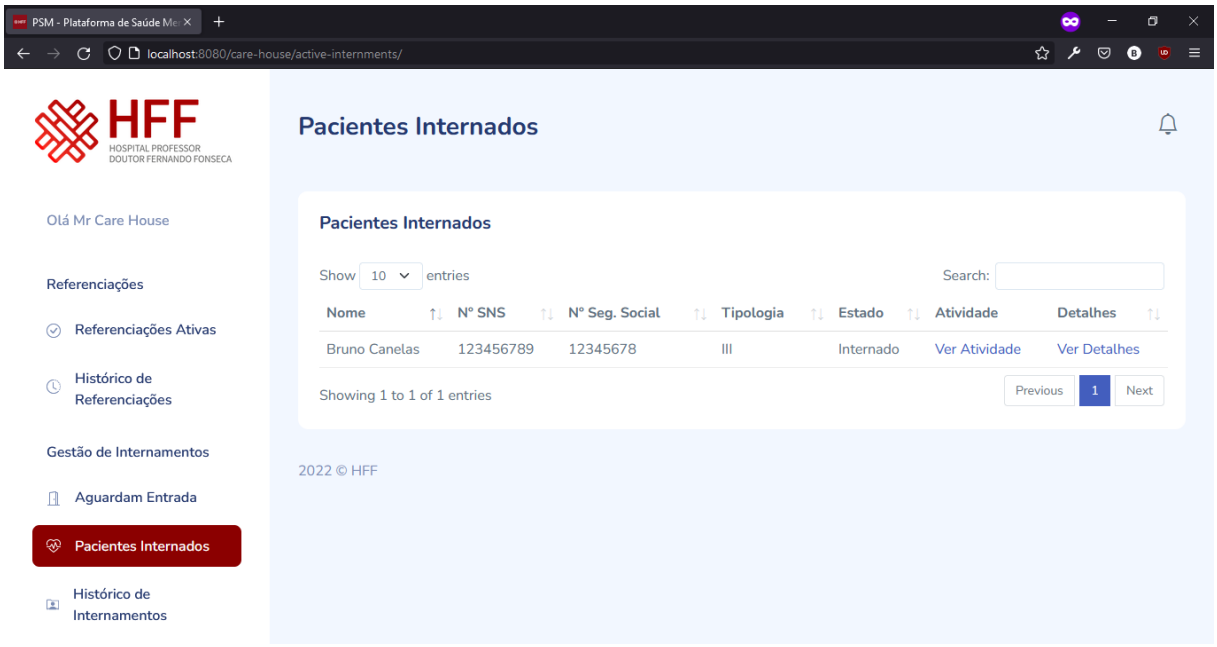

Figure 4.25: Care House - Active Internment

Once a patient has been admitted into the care house, they will then appear in the Care House Active Internment page, displayed in Figure 4.25. In this list the user is able to check the data of each patient that is interned in the respective care house, as well as visualize the patient's activity and add activity logs entries to the patients internment.

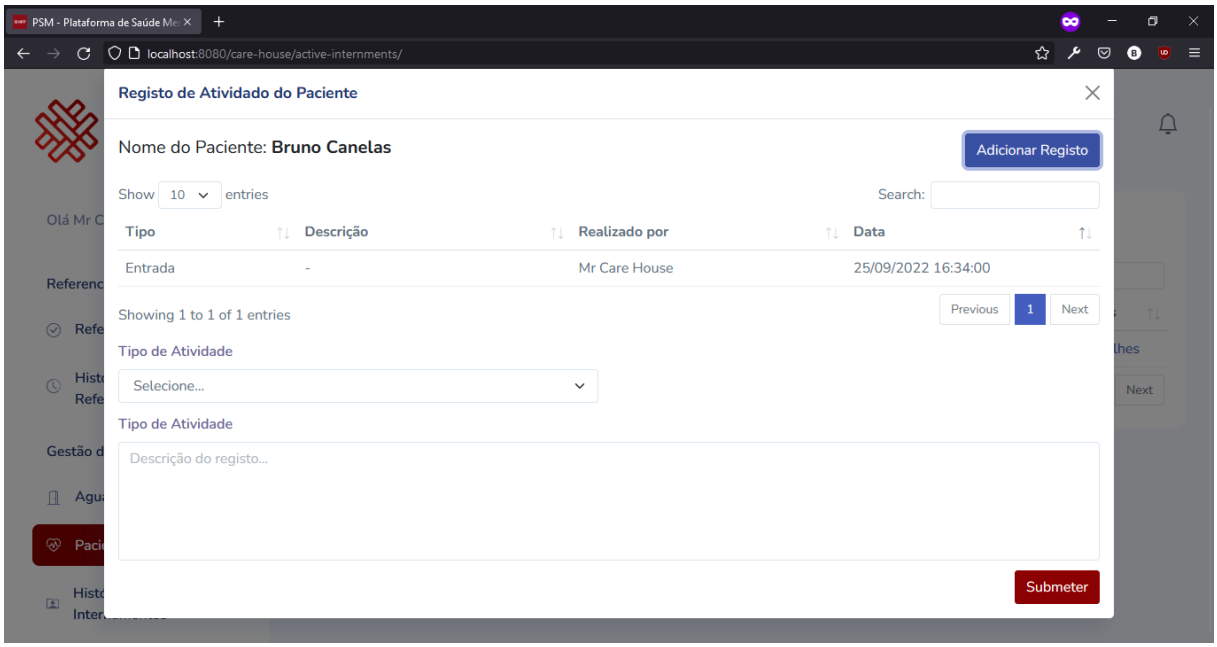

Figure 4.26: Care House - Activity Logs

If the user chooses to verify the patients activity, the pop-up displayed in Figure 4.26

will appear on the page. In this pop-up the user if able to check the entire activity history for the patient's internment as well as add a new activity log.

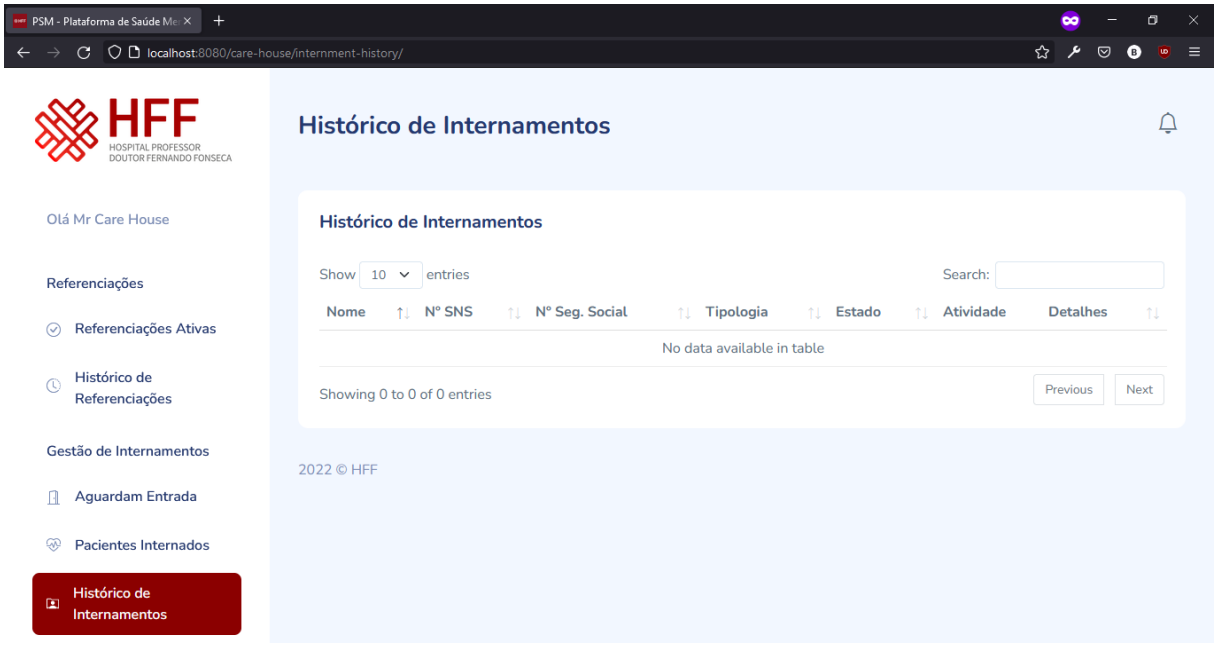

Figure 4.27: Care House - Internment History

Once the patient's internment is terminated, the internment will then be moved to the Care House Internment History page shown in Figure 4.27. In this page the user will be able to see the list of internments that have been concluded, along with the respective activity history. The user cannot change the data on this page, nor create new records.

The pages displayed above are part of the Internment Management section in the Care House Staff user type dashboard. These pages were based on the mock-ups shown in a previous chapter, however some adjustments have been made in order to improve the readability and organization of the page. These changes have mostly been done in regard to the activity logs. In the mock-ups, the access to the activity logs was made in the patient details pop-up, however by moving the option to the listing of patients it facilitates the access to this information. Another change that was made was the process for adding a new activity log. In the created mock-ups this process was done by adding a new line to a table, however, by creating a from in the bottom of pop-up, it makes the visualization clearer to the user that is adding the new log.

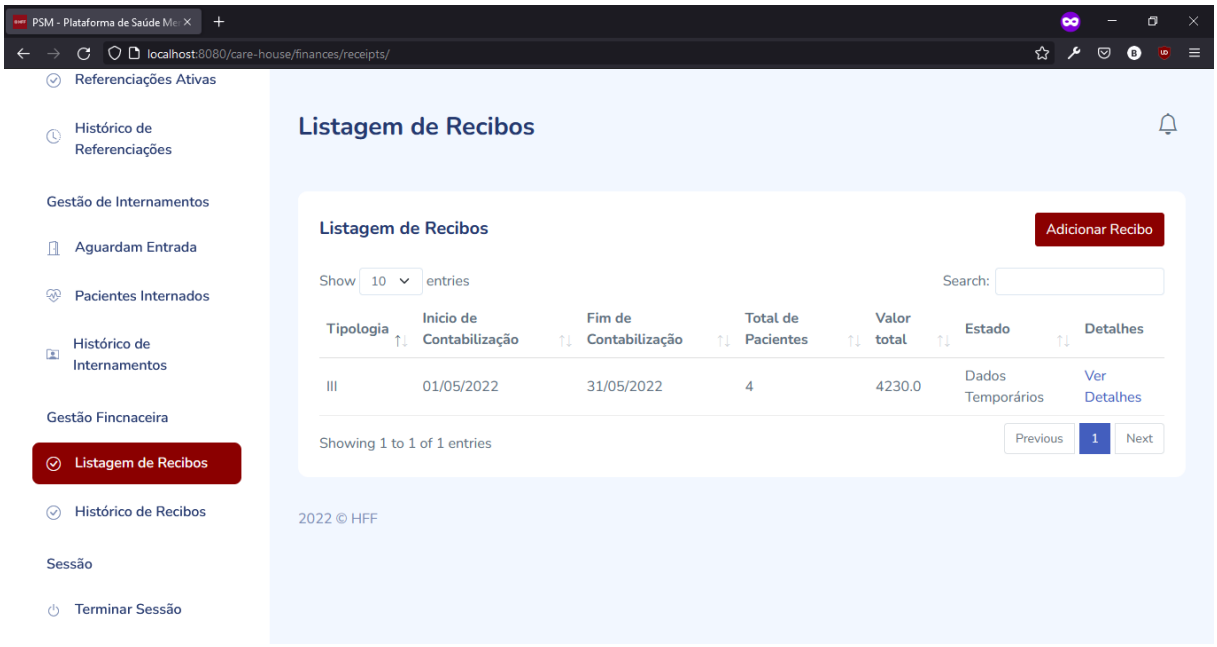

Figure 4.28: Care House - Active Invoices

Figure 4.28 represents the Care House Active Invoices Page. This page contains the list of receipts that have been created and still have temporary data or are yet to be reviewed by the Financial Staff.

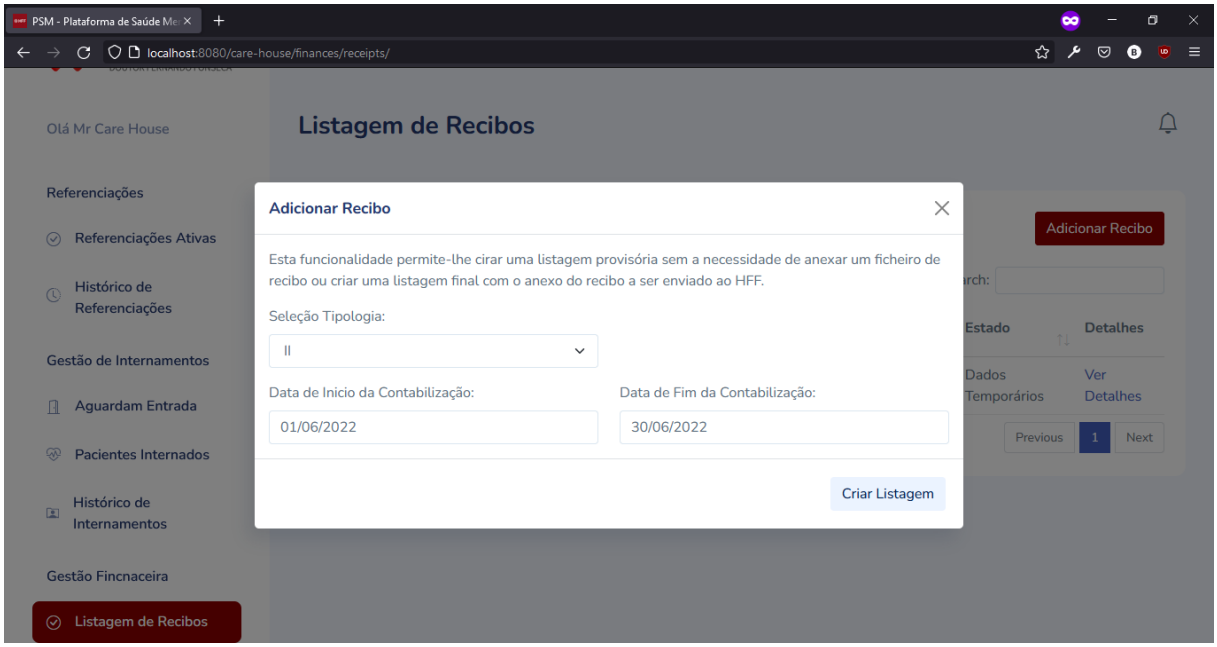

Figure 4.29: Care House - Add Invoice

Once the user wants to add a new invoice the pop-up shown in Figure 4.29 will be displayed in the page. The user must then select one of the typologies to create a new

invoice listing. If the previous invoice is still being processed the system will not allow the user to create a new invoice. Once the selection has been made the system will display the start and end dates of the invoice to the user. Once the user submits the data, the invoice listing will be created and added to the active invoices list.

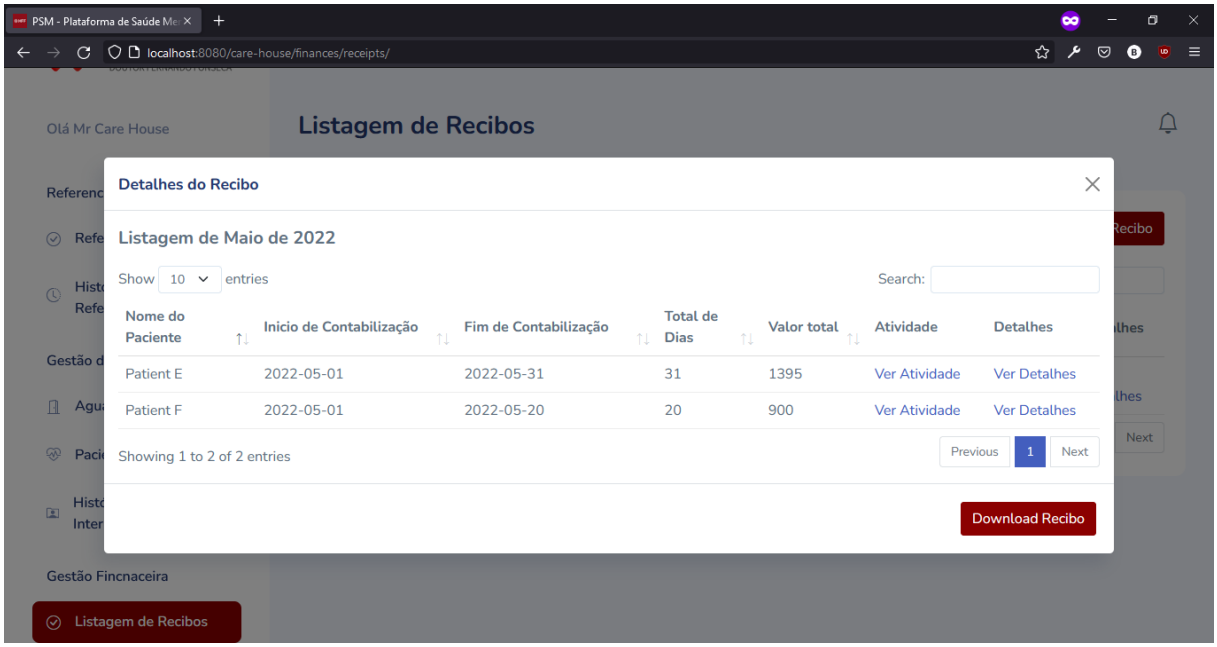

Figure 4.30: Care House - Invoice Details

By checking the invoice details, the user will be shown the pop-up displayed in Figure 4.30. In this pop-up the user will be able to verify the list of patients that will be accounted for in the receipt, with the number of days that each patient will be charged for and the start and end dates of the charge. The user is also able to verify the activity history for the patients and the details of their internment. If the invoice listing is still in a temporary status, the user will be able to submit an invoice file in order to finalize the receipt and send it for evaluation by the Financial Staff. If the the invoice file has already been attached to the listing, the user will then be able to download the file.

Figure 4.31: Care House - Invoice History

Once the invoice has been evaluated by the Financial Staff it will then be moved to the Care House Invoice History page shown in Figure 4.31. This page contains the list of invoices that have already been evaluated, either they have been accepted or rejected. The user cannot change or add records, however the invoice file can be downloaded from the invoice details pop-up.

The pages displayed in this subsection represent the Financial Management section in the Care House dashboard. The pages were created based on the mock-ups presented in Chapter 2 however they were optimized to allow the user to visualize the data with a better organization. The differences are primarily noticed in the ability to add a new invoice where in the mock-ups the typology selector was not present, being the invoices all generated at the same time. In the web application's page the invoices are generated individually in order to provide the user with better control over the generated data. If one of the invoices is rejected, this way the user will be able to generate the listing for that invoice in an isolated form.

4.2.4.4 Financial Staff

The Financial Staff users are employees of the HFF that are responsible for the management of the invoices generated by the Care Houses. These users also have access to some statistical data to help the HFF better prepare for future internments.

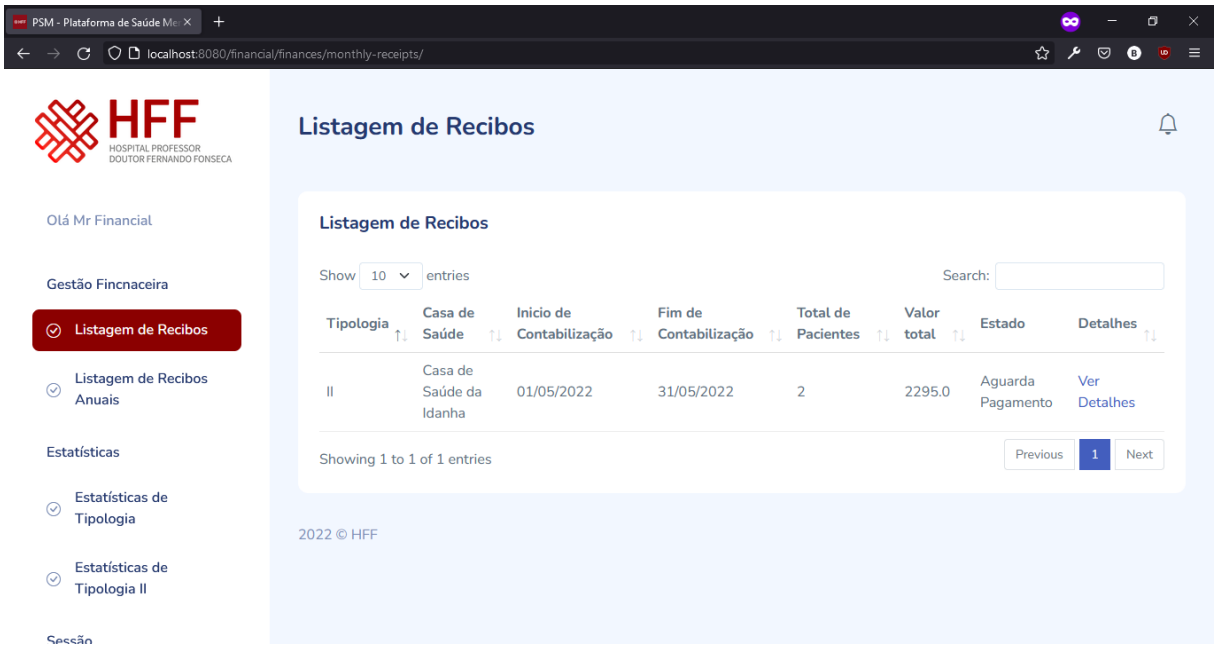

Figure 4.32: Financial - Active Invoices

Figure 4.32 represents the Financial Staff Active Invoices page. In this page the Financial Staff can verify the final invoices that were generated by the care houses and evaluate them in order to pay the amount indicated in the invoices. If the invoice is accepted, it will then be paid to the care house, otherwise the care house will need to generate a new invoice.

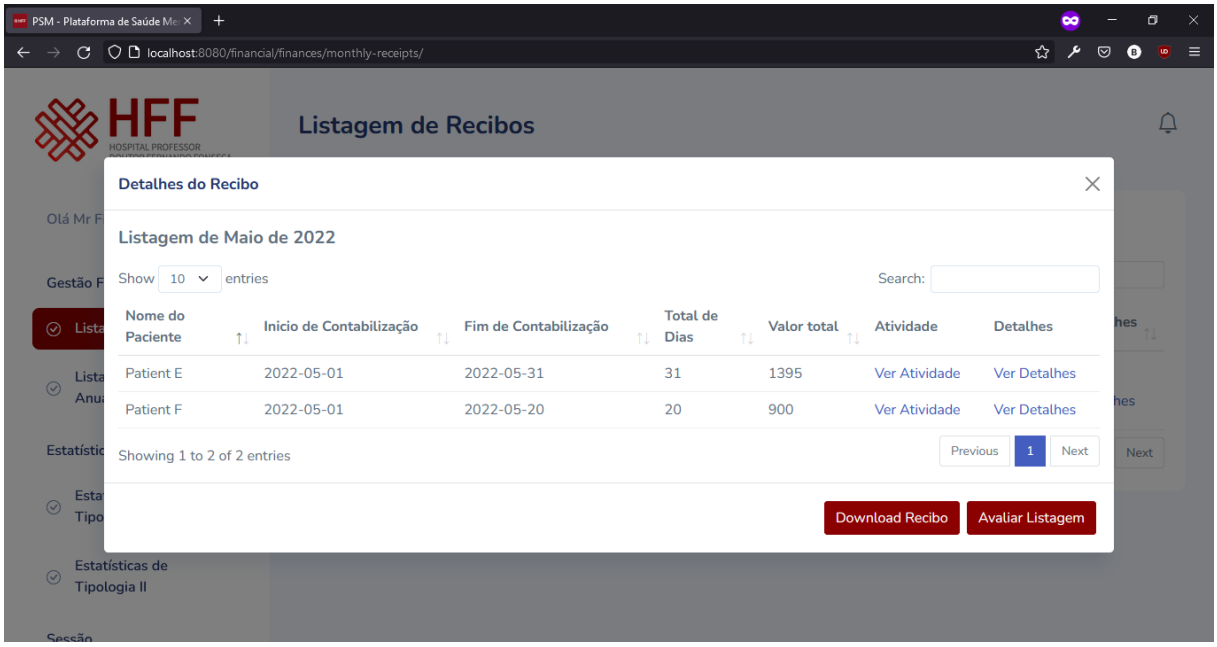

Figure 4.33: Financial - Invoice Details

The user can also check the invoice details, to which the pop-up shown in Figure 4.33 will appear. The user can evaluate the invoice from this pop-up or download the invoice file.

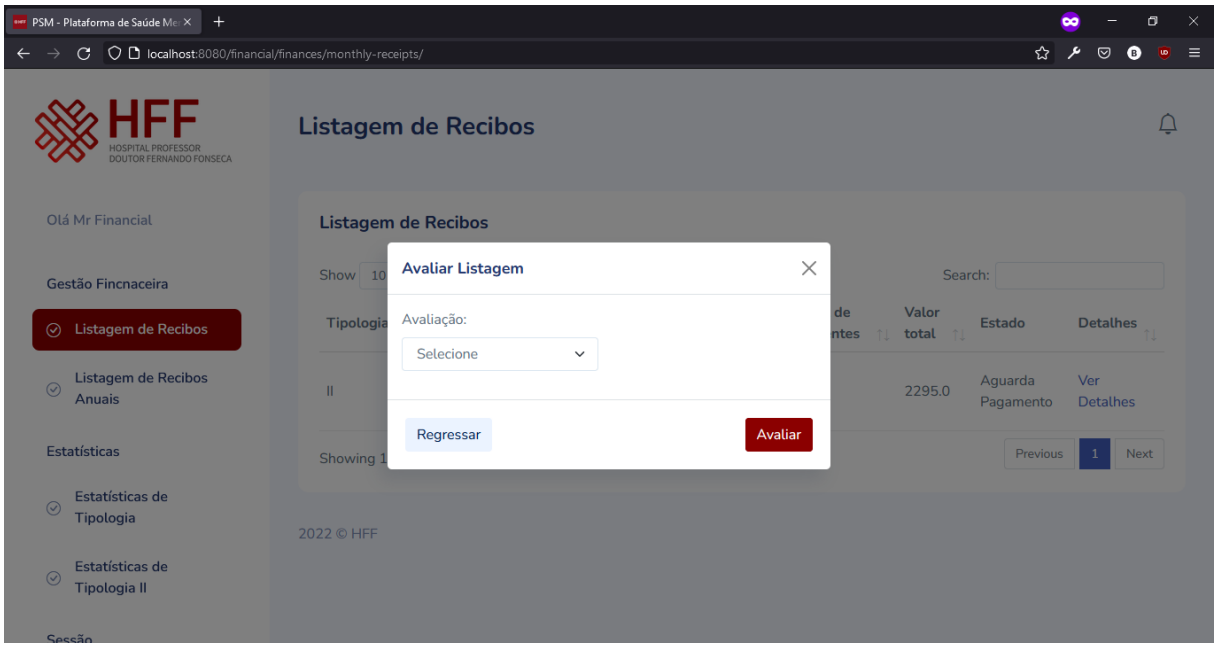

Figure 4.34: Financial - Invoice Evaluation

Once the user is ready to make the evaluation of the invoice, the pop-up displayed in Figure 4.34 will appear. The user will then need to indicate if the invoice is accepted or rejected.

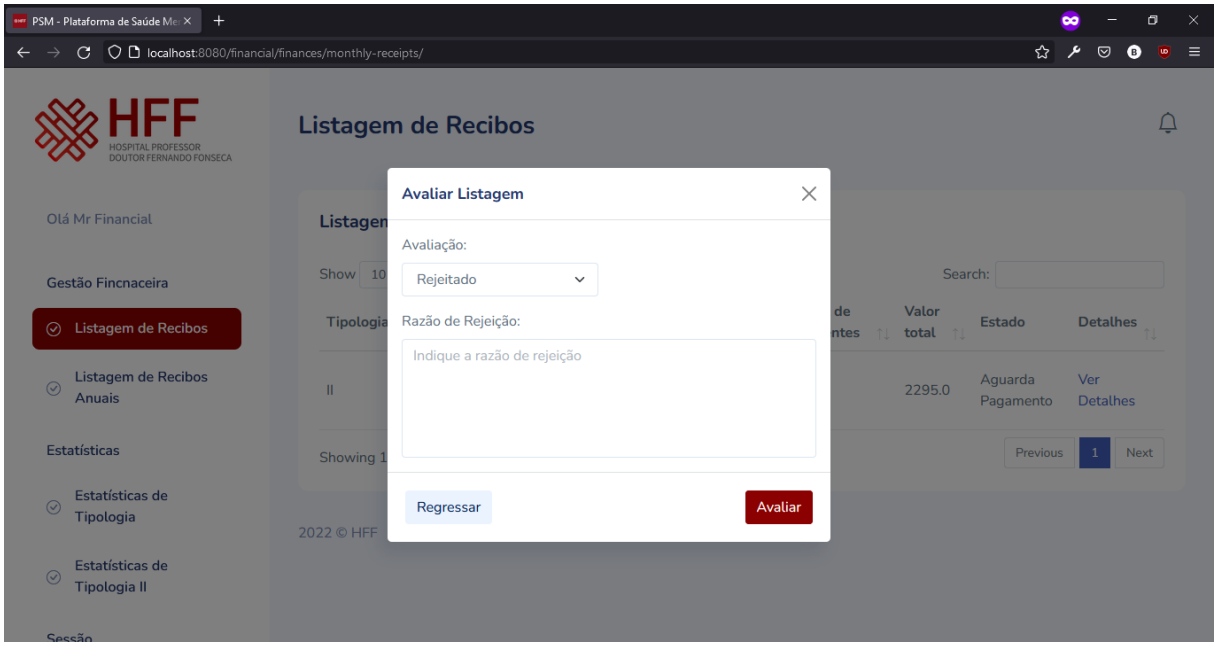

Figure 4.35: Financial - Invoice Evaluation Rejection

If the user decides to reject the invoice, then the pop-up will display a new field for the rejection reason, as shown in Figure 4.35.

Once the evaluation is completed, the invoice will then be moved to the invoice history page.

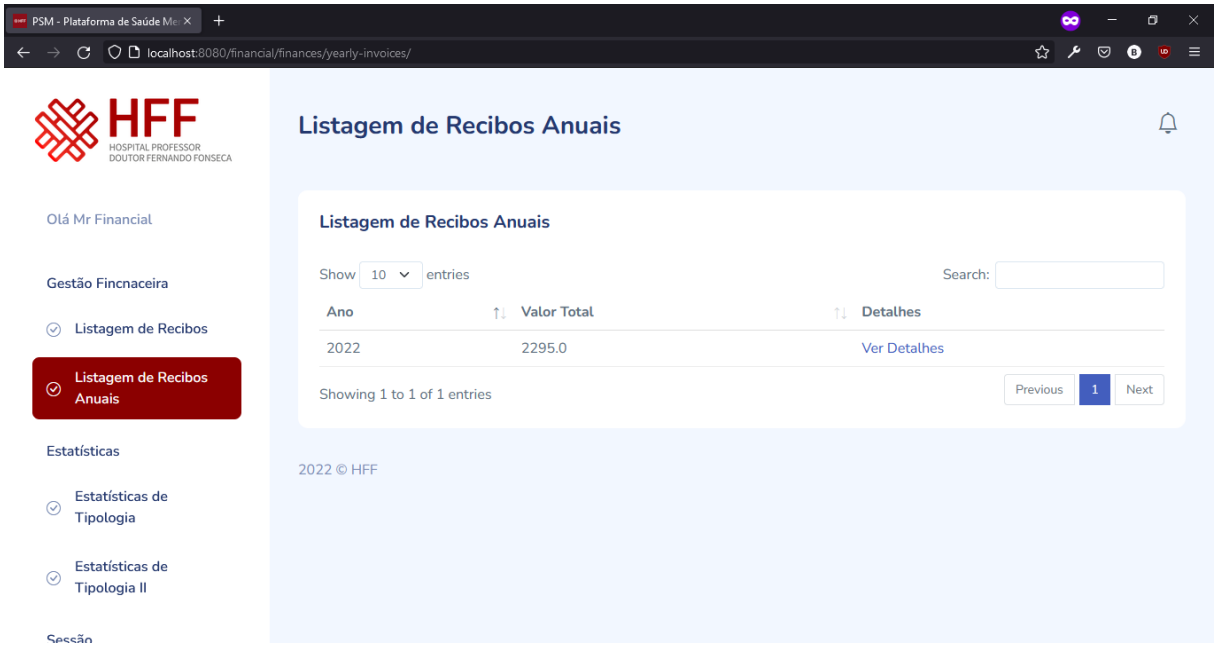

Figure 4.36: Financial - Yearly Invoices

Besides the monthly invoices, the user is also able to verify the data present in the yearly invoices, displayed in the page shown in Figure 4.36. These invoices are only created or updated once the monthly invoices are accepted.

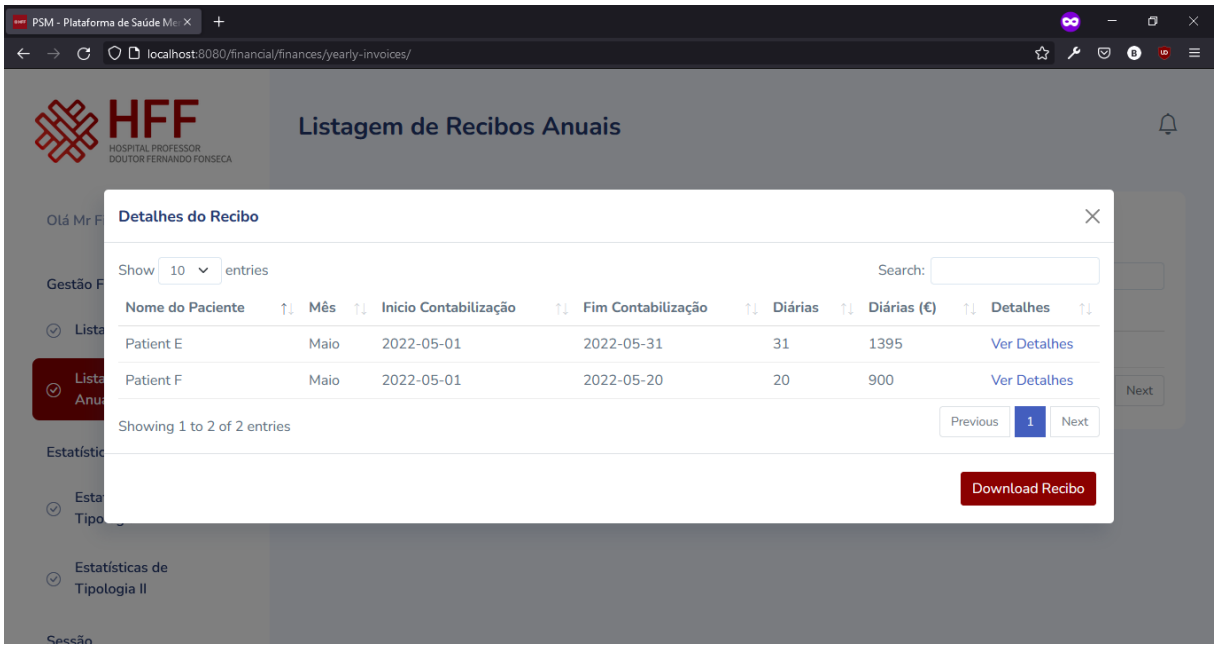

Figure 4.37: Financial - Yearly Invoices Details

By checking the yearly invoice details, the user is displayed with the pop-up represented in Figure 4.37. The details of the invoice show the list of patients and what they will be charged for each month, alongside with the number of days charged in a certain month and the total amount for that month. Data in this page cannot be changed, however the user can download a copy of the data in the page into a XLSX file.

The pages displayed in this subsections are part of the Financial Management section in the Financial Staff dashboard. These pages were based on the mock-ups indicated in chapter 2, however some changes were made in order to have a better organization of the page's contents. The changes were indicated in the previous user type.

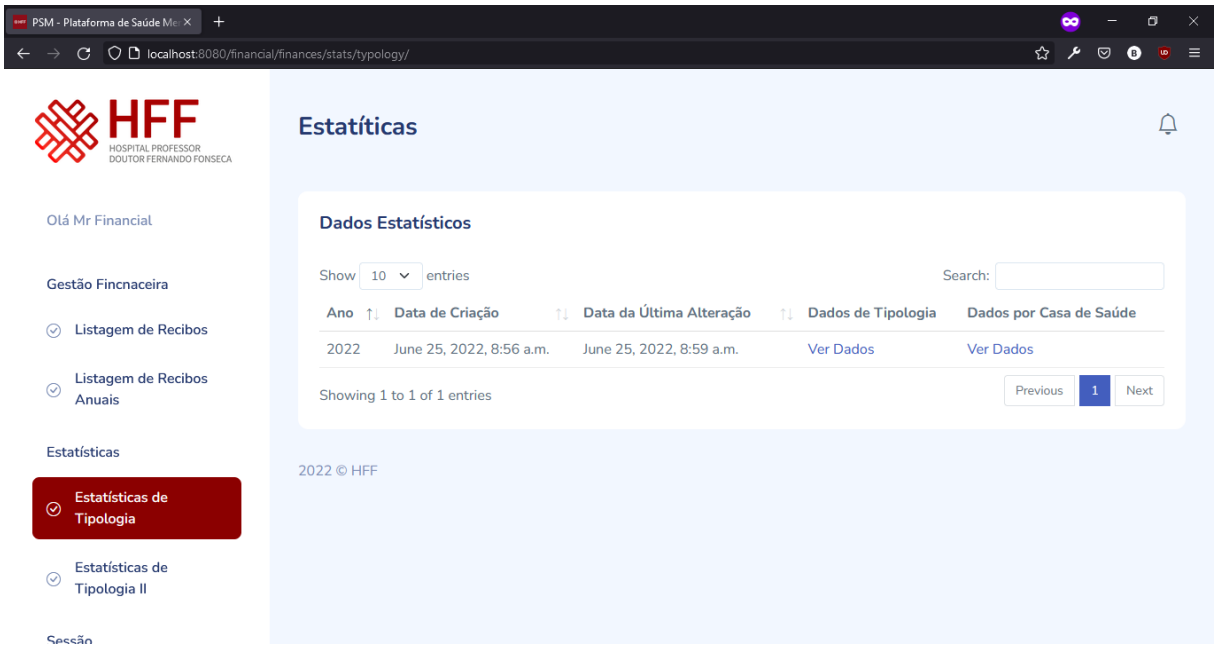

Figure 4.38: Financial - Care House Stats

Figure 4.38 represents the Financial Care House Stats page. This page displays the list of yearly care house stats that are useful for the HFF staff.

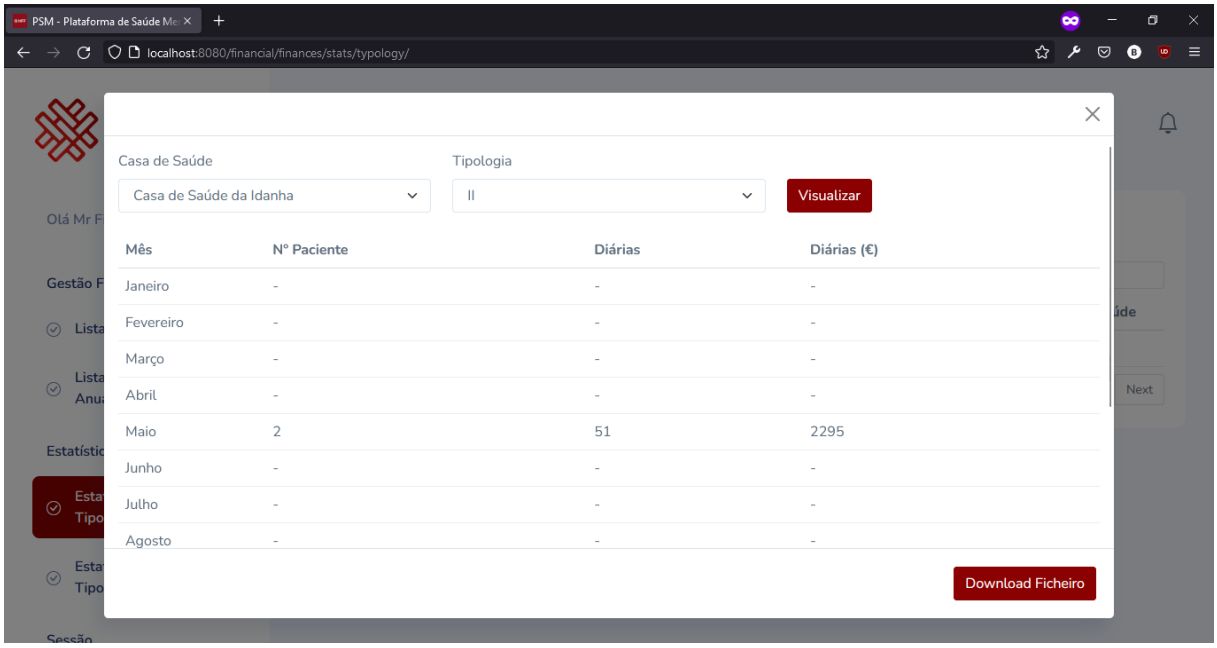

Figure 4.39: Financial - Care House Stats Details

Viewing the details of the care house stats will display the pop-up shown in Figure 4.39. In this pop-up the user is able to select the care house which they want to verify

the data for and which typologies are to be verified. This will then show the table with the data divided by month. The user can then download the table into a XLSX file.

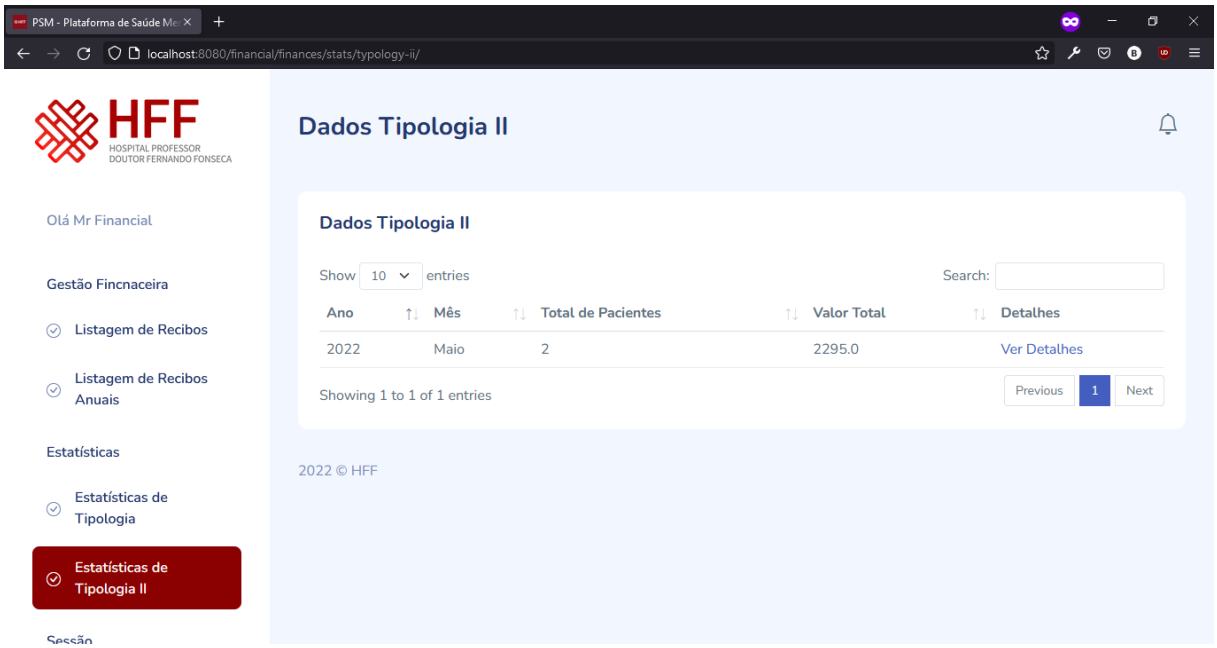

Figure 4.40: Financial - Typology II Stats

Besides the care house stats, the user is also able to verify the list of other institutions that have sent patients to be taken care by the HFF care houses and the amount of money they owe in any given month. This can be verified in the page represented by Figure 4.40. The user is also able to download a copy of this data into a XLSX file.

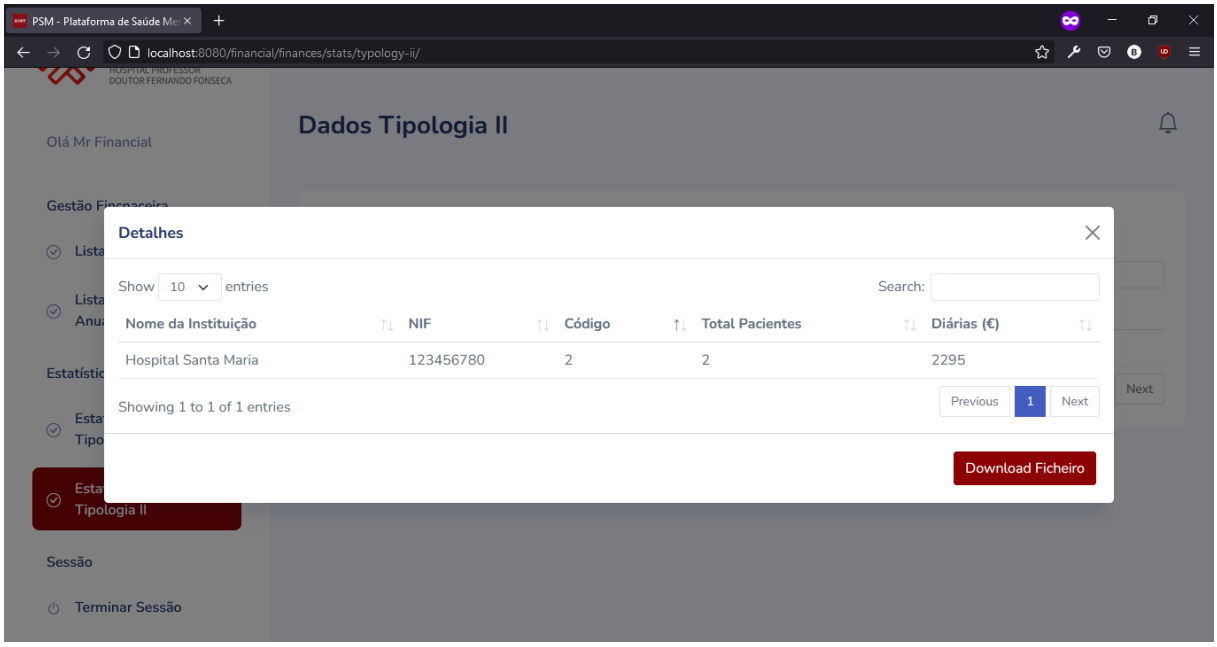

Figure 4.41: Financial - Typology II Stats Details

By clicking to check the details, the pop-up shown in Figure 4.41 will be displayed. The user can verify how much each individual institution owes and the amount of patients for each one. The user is also able to make a copy and download it into a XLSX file.

The pages displayed in this subsection are part of the Statistics section in the Financial Staff dashboard. The pages were based on the mock-ups indicated in Chapter 2, however in order to make a better visualization of the data, the tables were changed to fit the screen of a personal computer. This allows for a better visualization without the need for extra scrolls to visualize the full data. When the data is downloaded to a file, the structure indicated in the mock-ups is kept.

4.2.4.5 Superuser

The superuser has access to all the pages displayed to the other users. This gives the Superuser the ability to perform any action over a referral or invoice. However the superuser does not have access to the Admin Panel, this because the superuser is a member of the HFF and the Admin Panel allows a user to perform actions that might be harmful to the processes involved in the management of the patients.

4.2.5 Testing the Project

In order to test the project, Django provides a set of libraries and utilities that allow the creation of unit tests and integration tests. The tests are created in each application's tests folder.

To create a set of tests, a class must be created and extend the "TestCase" class provided by the Django Framework. After the class is created, the developer must then create the test methods, where the name of the method must start with "test_". The tests are run in alphabetic order, both the classes and the methods. When the tests are run, the Django Framework creates a test database with the same configuration as the default database. This test database is then deleted once the tests are finished.

In order to test the interactions with the Hosix system, a separate database was created using SQLite. This database contains all the fields that are present in the Hosix system in order to replicate the system. To select if the project will use the production Hosix system or the test system, a variable was created in the settings file. This variable is affected in the tests in order to use the test system.

To test the project, a set o integration and unit tests have been developed in order to simulate the actions made in the web application. The tests were divided between the

applications in order to test the specific aspects of each application. The created testes were the following:

- internment_management: For the Internment Management Application, 2 sets of tests were created, one for the referral process and another for the internment process. These sets were created in two files, test_a_referrals and test_b_internments. The tests created for the referrals process serve as a way to simulate the creation of a referral, until the moment the patient is interned in the care house. This process includes the creation of the referral by a Doctor, the approval by the care house, followed by the approval of the Reviewer and finally the indication that the patient has been interned in the care house. Some other tests were created in order to verify if the user was receiving the correct response for the actions that were being taken, for example rejecting a referral, trying to approve a referral that does not exist or changing the state of a referral that has already been approved or rejected. For the internment process, the tests created involved the actions taken by the Care House Staff when patient goes to an external consultation or the internment process is terminated. The tests created involved creating activity logs for external consultations, returns to the care house and terminating the process, all while verifying the correct state transitions. There was also the creation of tests that verified incorrect actions, for example trying to change the state of a terminated process or changing the internment into an invalid state.

- financial: For the Financial Application, the set of tests that was created served as a verification for the correct creation of invoices and their approval. These tests were created in the file test_invoices. The tests involved the Care House Staff creating and invoice and finalizing it, followed by the Financial staff approving the invoice. Some other scenarios were also tested, for example, trying to create an invoice while one was being processed, finalizing an already final invoice and approving or rejecting a temporary invoice.

To run the created tests, the following command must be run in the command line:

```
python manage.py test <app_name>.tests.<file_name>
```

This command will run all the testes created inside the given file and display the result to the user.

4.2.6 Deploying the Application

The deployment of the project was made in two instances: one for testing and another to simulate a production environment. The testing instance is a Heroku instance (free tier) where the deployment was made automatically and it is managed by the Heroku system. The production simulation deployment was made manually and requires certain dependencies in order to work correctly.

Both deployments require the "gunicorn" library in order to be deployed. In the testing environment the project is deployed using sync workers since that is the allowed configuration by Heroku. On the production simulation, the workers have been defined as async in order to improve the performance and increase the number of possible requests that can be made in concurrency.

To deploy the application in the production simulation, it was necessary to create a set of services in order to ensure the correct deployment of the project. The first service was the gunicorn service, which will allow the application to be executed in production mode. This service requires two files: psm_gunicorn.socket and psm_gunicorn.service

```
# psm_gunicorn.socket
[Unit]
Description=psm gunicorn socket

[Socket]
ListenStream=/run/psm_gunicorn.sock

[Install]
WantedBy=sockets.target
```

Figure 4.42: Gunicorn Socket definition

The first file defines the path and name of the socket that will listen to requests and direct them to the project.

```
# psm_gunicorn.service
[Unit]
Description=psm gunicorn daemon
Requires=psm_gunicorn.socket
After=network.target

[Service]
User=user
Group=root
WorkingDirectory=/home/user/2021_2022_tfm_psm
ExecStart=/home/user/2021_2022_tfm_psm/env/bin/gunicorn \
        --access-logfile - \
        -k uvicorn.workers.UvicornWorker \
        --workers 4 \
        --bind unix:/run/psm_gunicorn.sock \
        psm_django.asgi:application

[Install]
WantedBy=multi-user.target
```

Figure 4.43: Gunicorn Service definition

The second file defines the service that will run the application. This file defines some options like the number of workers for the project, the worker timeout among other settings.

After defining the service that will run the project, it is then necessary to configure Nginx to redirect the requests to the process. To do this, it is necessary to add an entry in the Nginx sites configuration file with the following parameters:

```
server {
    listen 80;
    server_name <server_domain_or_IP>;

    location = /favicon.ico {
        access_log off; log_not_found off;
    }
    location /static/ {
        root /home/user/2021_2022_tfm_psm;
    }

    location / {
        include proxy_params;
        proxy_pass http://unix:/run/psm_gunicorn.sock;
    }
}
```

Figure 4.44: Nginx Configuration

The configuration indicates the location of the socket to redirect the requests to, along with the port that will be listening for requests and the domain name that listens to the requests.

After configuring the indicated files, the project is ready to run and can be accessed using the indicated domain.

5

Conclusions

The project presented in this document was designed and developed according to the guidelines provided by HFF. Throughout the creation of the project, the members of the hospital provided feedback to the ideas and solutions presented to them, aiming to achieve the correct implementation of the requirements indicated.

During the development of the project, the members of the HFF provided the necessary clarifications in order for the project to better fit their necessities as an organization. The management of mental health patients is a complex process with multiple steps and requirements that need to be fulfilled to ensure the patients have the best care possible. To guarantee that this process was possible to replicate in an integrated management application, the members of the HFF provided files and real-life examples that were used in order to develop the project with the intent of facilitating the management process.

The integration with the Hosix system, which allows the search and creation of the patients in the HFF database, was provided by the members of the hospital and allows the project to have a direct link with the necessary data to provide the users with more detailed information about the patients.

5.1 Main Contributions

With the creation of this project the hospital has now access to an integrated management application that will allow the different departments involved in the patient

management to communicate in a faster and more efficient way, without the need of loose files, since everything is stored in a centralized database for them to access. The use of the developed project will make the referral and internment processes more reliable, without the risk of losing the emails flow or the insertion of the data in multiple locations.

Integrating the project with the Hosix system allows the users to create processes in a centralized manner since the insertion of the data in the web application will also allow the insertion of the same data in the HFF database. The users will also be able to search for patient data with the guarantee that the data is provided by the HFF, if the records for patient exist.

5.2 Future Work

The project was developed as a pilot project with the intent to expand to other organizations. The CNSC (in Portuguese "*Coordenação Nacional de Saúde Menatal*") is in charge of 5 regional organizations and the project was developed with the goal of creating an application that could be used by all the organizations. This project was a prototype developed only to be used by the HFF. For the future, the project is intended to be expanded to the regional organization of Lisbon and later to the remaining organizations. For this, some changes need to be made to ensure the consistency of the data and correct functioning of the application.

In order to guarantee a more consistent set of data regarding the patients, the integration with the Hosix system can also be refactored. The integration with this system was made in the ending phase of the project's development due to delays and constraints with the creation of the WebService by the HFF. When the WebService was then introduced, some of the fields that were not contemplated in the initial requirements and needed to be added to the project in order to make sure the integration worked with success. As such, some of the non-required fields were not added, which can lead to some incomplete sets of data.

References

[1] Filippo Ricca Gianna Reggio Maurizio Leotta and Diego Clerissi, "Dusm: A method for requirements specification and refinement based on disciplined use cases and screen mockups", 2018.

[2] G. Van Rossum and P.D. Team, *Learning Python: Crash Course Tutorial*. SPRINGER NATURE, 2020, ISBN: 9785304117357. [Online]. Available: https://books.google.pt/books?id=JG3BzQEACAAJ.

[3] S. Linge and H.P. Langtangen, *Programming for Computations - Python: A Gentle Introduction to Numerical Simulations with Python 3.6* (Texts in Computational Science and Engineering). Springer International Publishing, 2019, ISBN: 9783030168773. [Online]. Available: https://books.google.pt/books?id=oCa7DwAAQBAJ.

[4] H.J. Schönig, *Mastering PostgreSQL 12: Advanced techniques to build and administer scalable and reliable PostgreSQL database applications, 3rd Edition*. Packt Publishing, 2019, ISBN: 9781838985271. [Online]. Available: https://books.google.pt/books?id=g1jBDwAAQBAJ.

[5] Ubuntu. "Ubuntu 20.04". (), [Online]. Available: https://releases.ubuntu.com/20.04/.

[6] Gunicorn. "Gunicorn documentation". (), [Online]. Available: https://docs.gunicorn.org/en/stable/index.html.

[7] Nginx. "Nginx documentation". (), [Online]. Available: http://nginx.org/en/docs/.

[8] WSGI. "Wsgi documentation". (), [Online]. Available: https://wsgi.readthedocs.io/en/latest/what.html.

[9] ASGI. "Asgi documentation". (), [Online]. Available: https://asgi.readthedocs.io/en/latest/.

[10] E. Newcomer, *Understanding Web Services: XML, WSDL, SOAP, and UDDI* (Independent technology guides). Addison-Wesley, 2002, ISBN: 9780201750812. [Online]. Available: https://books.google.pt/books?id=M1rADzqjo2cC.

[11] "Iso 3166 country codes". (), [Online]. Available: https://www.iso.org/iso-3166-country-codes.html.

[12] Django Extensions. "Django extensions documentation". (), [Online]. Available: https://django-extensions.readthedocs.io/en/latest/.

[13] Wikipedia. "Mustache (template system)". (), [Online]. Available: https://en.wikipedia.org/wiki/Mustache_(template_system).

[14] Fred B. DuFresne, *Client-server system using embedded hypertext tags for application and database development*, 1996. [Online]. Available: https://patents.google.com/patent/US5835712A.

Documentação Hosix

WebService para o obtenção e criação dos dados dos processos no Hosix.

Será disponibilizado WebService chamado SearchPatients para a obtenção de dados dos processos de Hosix.

Na chamada ao WebService poderão ser passados os seguintes parâmetros de pesquisa:

InternalNumber – Identificador do processo no Hosix.
SnsNumber – Numero do Cartao de Utente (SNS)
PatientName – Nome do doente.
BirthDate – Data de nascimento do doente. (sem horas)
Sex – Sexo do doente

O InternalNumber e o SnsNumber obtem um único doente já que são campos chaves.

Para a pesquisa pelos restantes campos recomenda-se realizar o filtro por PatientName, BirthDate e Sex de modo que a pesquisa seja mais especifica.

Sera alterada a pesquisa de modo que a pesquisa por nome seja realizada por "Contains" e não pelo nome completo como esta a ser realizada o Alert. De iste modo não tera que coincidir o nome por completo para que a pesquisa devolva dados.

Será disponibilizado WebService chamado CreatePatient para a criação no Hosix de Processo de Doente.

Dados que serão processados pelo WebService

InternalNumber – Numero de processo
SnsNumber – Numero de Cartao de Utente
PatientName – Nome do doente
BirthDate –Data de Nascimento
Sex – Sexo do doente
Address – Endereço
PostalCode – Codigo Postal
Location - Localidade
Country - Pais
Nationality - Nacionalidade
FatherName - Nome do Pai
MotherName –Nome da Mae
Exemption – Codigo da Isenção
ExemptionDesc – Descritivo da Isenção
MaritalState – Estado Civil
SubsystemCode – Codigo da Entidade
SubsystemDesc – Descritivo da Entidade
BeneficiaryNumber – Numero Beneficiario
Profession - Profesion
DocumentationType – Tipo de documento
Documentation – Numero do Documento
PhoneNumbers – Telefone (array podem ser carregados até 4 telefones)
Email - Email

6 ANEXOS

6.1.1 Sexo do paciente

Descrição	Valor
Feminino	F
Masculino	M
Indeterminado	I

6.1.2 Países e nacionalidades

A codificação usada para os países usa a norma ISO 3166-1 (códigos alpha-2).

Descrição	Valor
<nenhum>	-1
Afeganistão	AF
África do Sul	ZA
Aland, Ilhas	AX
Albânia	AL
Alemanha	DE
Andorra	AD
Angola	AO
Anguila	AI
Antárctida	AQ
Antígua e Barbuda	AG
Antilhas Neerlandesas	AN
Arábia Saudita	SA
Argélia	DZ
Argentina	AR
Arménia	AM
Aruba	AW
Austrália	AU
Áustria	AT
Azerbaijão	AZ
Baamas	BS
Bangladeche	BD
Barbados	BB
Barém	BH
Bélgica	BE
Belize	BZ
Benim	BJ
Bermudas	BM
Bielorrússia	BY
Bolívia	BO

Bósnia e Herzegovina	BA
Botsuana	BW
Brasil	BR
Brunei Darussalam	BN
Bulgária	BG
Burquina Faso	BF
Burundi	BI
Butão	BT
Cabo Verde	CV
Caimão, Ilhas	KY
Camarões	CM
Camboja	KH
Canadá	CA
Catar	QA
Cazaquistão	KZ
Chade	TD
Chile	CL
China	CN
Chipre	CY
Cocos, Ilhas	CC
Colômbia	CO
Comores	KM
Congo	CG
Congo, República Democrática do	CD
Cook, Ilhas	CK
Coreia, República da	KR
Coreia, República Democrática Popular da	KP
Costa do Marfim	CI
Costa Rica	CR
Croácia	HR
Cuba	CU
Dinamarca	DK
Domínica	DM
Egipto	EG
Emirados Árabes Unidos	AE
Equador	EC
Eritreia	ER
Eslováquia	SK
Eslovénia	SI
Espanha	ES
Estados Unidos Da América	US
Estónia	EE
Etiópia	ET

Falkland (Malvinas), Ilhas	FK
Faroé	FO
Federação Russa	RU
Fiji	FJ
Filipinas	PH
Finlândia	FI
Formosa (Taiwan)	TW
França	FR
Gabão	GA
Gâmbia	GM
Gana	GH
Georgia	GE
Geórgia do Sul e Sandwich do Sul, Ilhas	GS
Gibraltar	GI
Granada	GD
Grécia	GR
Gronelândia	GL
Guadalupe	GP
Guam	GU
Guatemala	GT
Guernsey	GG
Guiana	GY
Guiana Francesa	GF
Guiné	GN
Guiné Equatorial	GQ
Guiné-Bissau	GW
Haiti	HT
Heard e McDonald, Ilhas	HM
Honduras	HN
Hong Kong	HK
Hungria	HU
Iémen	YE
Ilha Bouvet	BV
Ilha de Man	IM
Índia	IN
Indonésia	ID
Irão, República Islâmica do	IR
Iraque	IQ
Irlanda	IE
Islândia	IS
Israel	IL
Itália	IT
Jamaica	JM

Japão	JP
Jersey	JE
Jibuti	DJ
Jordânia	JO
Kuwait	KW
Laos, República Popular Democrática do	LA
Lesoto	LS
Letónia	LV
Líbano	LB
Libéria	LR
Líbia, Grande Jamahiriya Socialista Popular Árabe da	LY
Listenstaine	LI
Lituânia	LT
Luxemburgo	LU
Macau	MO
Macedónia, Antiga República Jugoslava da	MK
Madagáscar	MG
Malásia	MY
Malavi	MW
Maldivas	MV
Mali	ML
Malta	MT
Marianas do Norte	MP
Marrocos	MA
Marshall, Ilhas	MH
Martinica	MQ
Maurícia	MU
Mauritânia	MR
Mayotte	YT
Menores Distantes dos Estados Unidos, Ilhas	UM
México	MX
Micronésia, Estados Federados da	FM
Moçambique	MZ
Moldávia, República da	MD
Mónaco	MC
Mongólia	MN
Monserrate	MS
Montenegro	ME
Myanmar	MM
Namíbia	NA
Natal, Ilha de	CX
Nauru	NR
Nepal	NP

Nicarágua	NI
Níger	NE
Nigéria	NG
Niue	NU
Norfolk, Ilha	NF
Noruega	NO
Nova Caledónia	NC
Nova Zelândia	NZ
Omã	OM
Países Baixos	NL
Palau	PW
Palestina	PS
Panamá	PA
Papuásia-Nova Guiné	PG
Paquistão	PK
Paraguai	PY
Peru	PE
Pitcairn	PN
Polinésia Francesa	PF
Polónia	PL
Porto Rico	PR
Portugal	PT
Quénia	KE
Quirguizistão	KG
Quiribati	KI
Reino Unido	GB
República Centro-Africana	CF
República Checa	CZ
República Dominicana	DO
Reunião	RE
Roménia	RO
Ruanda	RW
Salomão, Ilhas	SB
Salvador	SV
Samoa	WS
Samoa Americana	AS
Santa Helena	SH
Santa Lúcia	LC
Santa Sé (Estado da Cidade do Vaticano)	VA
São Cristóvão e Nevis	KN
São Marino	SM
São Pedro e Miquelon	PM
São Tomé e Príncipe	ST

São Vicente e Granadinas	VC
Sara Ocidental	EH
Seicheles	SC
Senegal	SN
Serra Leoa	SL
Sérvia	RS
Singapura	SG
Síria, República Árabe da	SY
Somália	SO
Sri Lanca	LK
Suazilândia	SZ
Sudão	SD
Suécia	SE
Suíça	CH
Suriname	SR
Svalbard e Jan Mayen	SJ
Tailândia	TH
Tajiquistão	TJ
Tanzânia, República Unida da	TZ
Território Britânico do Oceano Índico	IO
Territórios Austrais Franceses	TF
Timor-Leste	TL
Togo	TG
Tokelau	TK
Tonga	TO
Trindade e Tobago	TT
Tunísia	TN
Turcas e Caicos, Ilhas	TC
Turquemenistão	TM
Turquia	TR
Tuvalu	TV
Ucrânia	UA
Uganda	UG
Uruguai	UY
Usbequistão	UZ
Vanuatu	VU
Venezuela	VE
Vietname	VN
Virgens Americanas, Ilhas	VI
Virgens Britânicas, Ilhas	VG
Wallis e Futuna	WF
Zâmbia	ZM
Zimbabué	ZW

6.1.3 Estado civil

Descrição	Valor
<nenhum>	-1
Casado	M
Divorciado	D
Separado de facto	F
Solteiro	S
União de facto	U
Viúvo	W

6.1.4 Profissão

Descrição	Valor
<nenhum>	-1
Agricultores E Pescadores-Agricultura E Pesca De Subsistência	27
Agricultores E Trabalhadores Qualificados Da Agricultura, Criação De Animais E Pescas	26
Agricultores E Trabalhadores Qualificados Da Agricultura E Pescas	6
Condutores De Veículos E Embarcações E Operadores De Equipamentos Pesados Moveis	34
Directores De Empresa	12
Directores E Gerentes De Pequenas Empresas	13
Docentes Do Ensino Secundário, Superior E Profissões Similares	16
Empregados De Escritório	22
Empregados De Recepção, Caixas, Bilheteiros E Similares	23
Especialistas Das Ciências Da Vida E Profissionais Da Saúde	15
Especialistas Das Ciências Físicas Matemáticas E Engenharia	14
Especialistas Das Profissões Intelectuais E Cientificas	2
Forcas Armadas	10
Manequins, Vendedores E Demonstradores	25
Mecânicos De Precisão, Oleiros E Vidreiros, Artesãos, Trab. Das Artes Gráficas E Trab. Similar	30
Operadores De Instalações E Maquinas E Trabalhadores Da Montagem	8
Operadores De Instalações Fixas E Similares	32
Operadores De Maquinas E Trabalhadores Da Montagem	33
Operários, Artificies E Trabalhadores Similares Das Indust. Extract. E Da Construção Civil	28
Operários, Artificies E Trabalhadores Similares	7
Outros Especialistas Das Profissões Intelectuais E Cientificas	17
Outros Operários, Artificies E Trabalhadores Similares	31
Outros Técnicos E Profissionais De Nivel Intermédio	21
Pessoal Administrativo E Similares	4
Pessoal Dos Serviços Directos E Particularidade Protecção E Segurança	24
Pessoal Dos Serviços E Vendedores	5
Profissionais De Nível Intermédio Do Ensino	20
Profissionais De Nível Intermédio Das Ciências Da Vida E Da Saúde	19
Quadros Superiores Da Administração Publica	11
Quadros Superiores Da Administração Publica, Dirigentes E Quadros Superiores De Empresa	1
Técnicos E Prof. De Nível Intermédio Das Ciências Fis. Químicas, Eng. E Trabalh. Similares	18
Técnicos E Profissionais De Nível Intermédio	3
Trabalhadores Da Metalurgia E Da Metalomecânica E Trabalhadores Similares	29
Trabalhadores Não Qual. Das Minas, Da Const. Civil E Obras Publi. Da Indust.Tran.E Transpor.	37
Trabalhadores Não Qualificados	9
Trabalhadores Não Qualificados Da Agricultura E Pescas	36
Trabalhadores Não Qualificados Dos Serviços E Comércio	35

6.1.5 Tipo de documento

Descrição	Valor
Bilhete de Identidade	1035
Carta de Condução	1031
Cartão de Cidadão	1054
Cartão de Segurança Social	1032
Cédula	1051
Cédula Militar	1052
Certidão de Nascimento	1049
NIF	1053
Passaporte	1034

6.1.6 Subsistemas de saúde

Descrição	Valor
ADMINISTRAÇÃO DO PORTO DE LISBOA	5017
ADMINISTRAÇÃO DOS PORTOS DOURO E LEIXÕES	5016
AEGON SEGUROS GENERALES,SA	5066
AIG	5079
ASSISTUR	5056
AXA PORTUGAL - COMPANHIA DE SEGUROS, SA	5024
BPA VIDA-COMPANHIA DE SEGUROS VIDA, SA	5028
C SEG A SOCIAL	5021
C SEG AIDE ASSISTENCE	5067
C SEG COMERC UNION (ALBA)	5078
C SEG COMMERC U ASSURANCE	5070
C SEG EAGLE STAR VIDA	5029
C SEG EURESAP	5092
C SEG GARANTIA	5052
C SEG GESA	5053
C SEG GUARDIAN	5034
C SEG GUARDIAN ASSURANCE	5072
C SEG INSURANCE	5073
C SEG INTER-ATLANTICO SA	5035
C SEG LA EQUITATIVA	5036
C SEG LA PRESERVATRICE	5074
C SEG LEGAL & GENERAL	5055
C SEG LIBERTY EUROPEIA	5098
C SEG MAAF	5058

C SEG METROPOLE SA	5038
C SEG MUTUELLE ASSURANCE	5075
C SEG NORTHERN	5090
C SEG O TRABALHO SA	5039
C SEG OCEANICA	5040
C SEG OK TELESEGURO	5102
C SEG PEARL PORTUGAL SA	5041
C SEG PHOENIX ASSURANCE	5077
C SEG PORTUGAL	5054
C SEG PRESERVATRICE FONCI	5080
C SEG ROYAL EXCHANGE	5043
C SEG ROYAL INSURANCE	5044
C SEG SCOTTISH UNION PORT	5046
C SEG SOCIED PORT SEGUROS	5061
C SEG SUN ALLIANCE PORTUG	5081
C SEG TAGUS SA	5047
C SEG TIARD PFA	5082
C SEG ULTRAMARINA SA	5049
C SEG UNION ASSURAN PARIS	5076
C SEG UNION FENIX	5069
C.A.SEGUROS	5100
COMPANHIA DE SEGUROS ASSICURAZIONI GENERAL SpA	5026
COMPANHIA DE SEGUROS AÇOREANA, SA	5023
COMPANHIA DE SEGUROS ALLIANZ PORTUGAL, SA	5042
COMPANHIA EUROPEIA DE SEGUROS, SA	5019
COMPANHIA PORTUGUESA SEGUROS SAUDE, SA	5020
COMPANHIA SEGUROS ABEILLE VIE-GROUPE COMMERCIAL UNION	5022
COMPANHIA SEGUROS ALICO-AMERICAN LIFE INSURANCE COMPANY	5025
COMPANHIA SEGUROS FIDELIDADE-MUNDIAL, SA	5030
COMPANHIA SEGUROS SAGRES,SA	5094
COMPANHIA SEGUROS TRANQUILIDADE, SA	5048
DIRECÇÃO-GERAL PROTECÇÃO SOCIAL AOS FUNCIONÁRIOS E AGENTES DA ADMINISTRAÇÃO PÚBLICA	5000
ESPAÑA, SA - COMPAÑIA NACIONAL SEGUROS	5051
ESPIRITO SANTO - COMPANHIA SEGUROS,SA	5097
EUROP ASSISTANCE - COMPANHIA PORTUGUESA SEGUROS DE ASSISTÊNCIA,SA	5071
GAN PORTUGAL SEGUROS, SA	5031
GENERALI C.SEGUROS,SpA	5101
GENESIS SEGUROS GENERALES,SOCIEDADANONIMA SEGUROS Y REASEGUROS	5032
GLOBAL - COMPANHIA DE SEGUROS,SA	5033
IBERO ASSISTENCIA SA	5093

IMA INTER MUTUELLE ASSIST	5087
IMPERIO BONANCA C SEG SA	5095
IMPERIO BONANÇA, COMPANHIA DE SEGUROS, SA	5027
IMPRENSA NACIONAL CASA MOEDA	5014
INSTITUTO ACÇÃO SOCIAL FORÇAS ARMADAS	5001
INSTITUTO SEGUROS DE PORTUGAL	5063
INSTITUTO NACIONAL DE SEGUROS	5062
LONDON GENERAL INSURANCE	5099
LUSITÂNIA COMPANHIA SEGUROS, SA	5037
MAPFRE SEGUROS GERAIS,SA	5068
MATMUT/SCOTTISH (CARES)	5088
MUTUAMAR-MÚTUA SEGUROS ARMADORES PESCA ARRASTO	5104
OCIDENTAL - COMPANHIA PORTUGUESA SEGUROS, SA	5059
PAIS DE ACORDO	5018
PFA SEGUROS	5064
RADIODIFUSAO PORTUGUESA	5015
REAL SEGUROS, SA	5060
RURAL SEGUROS-COMPANHIA SEGUROS RAMOS REAIS, SA	5091
SAD MUNICIPAL CM LISBOA	5004
SAD MUNICIPAL CM PORTO	5005
SCOTTISH UNION(LA FRANCE)	5086
SEG LUSO ATLANTICA SA	5057
SEG MARSH & MCLENNAN LDA	5083
SEGURO DIRECTO GERE-COMPANHIA SEGUROS,SA	5045
SEGUROS LLOYDS	5084
SEGUROS LOGO SA	5103
SERVAIDE-ASSIST SERVICOS	5096
SERVIÇO ASSISTÊNCIA DOENÇA - POLICIA SEGURANÇA PUBLICA	5003
SERVIÇO ASSISTÊNCIA DOENÇA - SERVIÇO ESTRANGEIROS E FRONTEIRAS	5006
SERVIÇO ASSISTÊNCIA DOENÇA AOS MILITARES DA GUARDA NACIONAL REPUBLICANA	5002
SERVIÇO ASSISTÊNCIA MEDICO-SOCIAL - QUADROS TECNICOS	5009
SERVIÇO ASSISTÊNCIA MEDICO-SOCIAL BANCÁRIOS CENTRO	5008
SERVIÇO ASSISTÊNCIA MEDICO-SOCIAL BANCÁRIOS NORTE	5007
Serviço Nacional Saúde	543
SERVIÇOS SOCIAIS CAIXA GERAL DEPOSITOS	5010
SERVIÇOS SOCIAIS MINISTÉRIO JUSTICA	5011
SERVIÇOS SOCIAIS TAP AIR PORTUGAL	5012
SERVIÇOS SOCIAIS TELEFONES LISBOA E PORTO	5013
SOC PORTUGUESA SEG (AGF)	5065
VAN AMEYDE PORTUGAL	5085

VICTORIA - COMPANHIA SEGUROS, SA	5050
ZURICH- COMPANHIA SEGUROS, SA	5089

6.1.7 Isenção

Descrição	Valor
Beneficiários do rendimento social de inserção	83
Militares e ex -militares das Forças Armadas que, em virtude da prestação do serviço militar, se encontrem incapacitados de forma permanente	108
Crianças até aos 12 anos de idade, inclusive	1002
Beneficiários de abono complementar a crianças e jovens deficientes	1003
Beneficiários de subsídio mensal vitalício	1004
Internados em lares para crianças e jovens privados do meio familiar normal	1008
Pensionistas de doença profissional com o grau de incapacidade permanente global não inferior a 50%	1010
Dadores benévolos de sangue	1012
Bombeiros	1017
Não isento	1018
Pensionistas que recebam pensão não superior ao SMN	5000
Pensionistas cujo rendimento do agregado familiar dividido por 2 é inferior ao SMN	5001
Dependentes de beneficiários dos códigos 0500 e 0501	5002
Desempregados inscritos nos Centros de Emprego	5003
Dependentes de Beneficiários do cód. 0601	5004
Beneficiários de Prestação de Caracter eventual por Sit. Carência paga por Ser. Oficiais	5005
Dependentes de beneficários do cód. 0701	5006
Trabalhadores por conta de outrem que recebam rendimento mensal não superior ao SMN	5007
Trabalhadores por conta de outrem cujo rendimento do agregado familiar dividido por 2 seja inferior ao SMN	5008
Dependentes de Beneficiarios dos cód.0900 e 0901	5009
Declaração Médica Temporária	5010
Declaração Médica Definitiva	5011