

Table of content

Table of figures.....	
Chapter 1. Introduction	1
1.1 Objectives and structure of the work	2
Chapter 2. Analysis and design of the computer system	3
2.1 System analysis	3
2.2 Designing the computer system	5
2.2.1 Conceptual data model	5
2.2.2 Logical data model	10
2.2.3 The physical model of data	13
Chapter 3. Description of the IT system	25
3.1 Administrator component	27
3.2 Receptionist Component	46
Chapter 4. Conclusions and proposals	59
Bibliography	60
Web References	61
Appendices	62

Table of figures

Figure 3.1. 1 Login window	28
Figure 3.1. 2 Login with wrong data	28
Figure 3.1. 3 Administrators menu	29
Figure 3.1. 4 Rooms management window	30
Figure 3.1. 5 Room category management window	31
Figure 3.1. 6 The "Beds" tab related to the selected category	32
Figure 3.1. 7 Warning message when adding an extra bed	32
Figure 3.1. 8 The "Utilities" tab related to the selected category	33
Figure 3.1. 9 The "Images" tab related to the selected category	33
Figure 3.1. 10 Image gallery of selected camera category	34
Figure 3.1. 11 Adding images to an image gallery	35
Figure 3.1. 12 The "images" file	36
Figure 3.1. 13 Services management window	36
Figure 3.1. 14 The selected service image setting form	37
Figure 3.1. 15 Rate options window	38
Figure 3.1. 16 Window for managing room rates	38
Figure 3.1. 17 The "Tariffs" tab related to the tariff management module	39
Figure 3.1. 18 Filtering the room rate history	39
Figure 3.1. 19 Window for the management of service tariffs	40
Figure 3.1. 20 Viewing the tariff history	40
Figure 3.1. 21 Form for the management of room discounts	41
Figure 3.1. 22 The form for the management of personal data	42
Figure 3.1. 23 The "Employee" option selected in the personal data management form ..	43
Figure 3.1. 24 Personal data management form with completed information	44
Figure 3.1. 25 Form for generating graphs	45
Figure 3.2. 1 Receptionist menu window	46
Figure 3.2. 2 Accommodation window opening form	46
Figure 3.2. 3 Accommodation form with reservation	47
Figure 3.2. 4 Information message about the availability of rooms	48
Figure 3.2. 5 Accommodation window	48
Figure 3.2. 6 Accommodation window	49
Figure 3.2. 7 Accommodation services selection page	50
Figure 3.2. 8 Customer registration form	51
Figure 3.2. 9 Form completed following the verification of CNP	51
Figure 3.2. 10 The information displayed on the last page of accommodation	52
Figure 3.2. 11 Booking window opening form	52
Figure 3.2. 12 Room selection page for reservation	53
Figure 3.2. 13 Final page of booking window	54

Figure 3.2. 14 Viewing the accommodation history	54
Figure 3.2. 15 Viewing the page for selected rooms in the history of an accommodation ..	55
Figure 3.2. 16 Viewing the page for the selected services in the history of an accommodation	55
Figure 3.2. 17 Viewing the page for customer data in the history of an accommodation ..	56
Figure 3.2. 18 17 Viewing the accommodation details page in the history of an accommodation	56
Figure 3.2. 19 Window for viewing a customer's reservations	57
Figure 3.2. 20 Viewing the page for selected rooms in the history of a reservation	57

Chapter 1. Introduction

In the past, hotels were local businesses that provided rooms and services to customers. In the early 1900s, the emergence of hotel chains began. This happened due to the high number of travelers and the high demand for accommodation. Because of this, and the fact that hotel chains expanded into different geographic locations, it was necessary to use a computer system, which would become a consistent method of tracking and maintaining hotel operations.

The information system represents the set of elements necessary for the process of collecting, transmitting and processing information. In a broad sense, the term is often used to refer to the interaction between people, processes, data and technologies. The term not only refers to the management of information, but also to how people interact with this technology.

The data is processed and the result can be transferred further to another processing compartment. This data transfer is carried out electronically. The elements involved in this data processing and transmission process constitute an IT system.

The computer system represents a set of electronic procedures and equipment that allow automatic data processing and obtaining information. System activities are intended to process, transmit, store, retrieve, manipulate, and display information in the most efficient, cost-effective, and error-free manner possible.

In the case of hotels, computer systems are used to keep track of accommodations, reservations, services provided, rates, and customers.

1.1 Objectives and structure of the paper

The main objective of the work consists in the design and implementation of an IT system for the management of room reservations in a hotel. During this work, several specific objectives of the IT system will be presented, the most representative being:

- customer accommodation, depending on the rooms available in the desired period;
- booking rooms and services for a specified period of time. This component will be accessible by hotel receptionists using the software application;
- view history for reservations and accommodations;
- management of discounts depending on the season and the category to which the customers belong;
- management of room rates and services that the hotel offers to customers;
- setting and changing representative images of services and rooms. This change can be made by administrators through the software application;
- making graphic reports according to the desired period.

Based on the previously stated objectives, the paper is structured in four chapters:

Chapter I presents introductory information regarding the process of accommodation and reservation of rooms and services;

Chapter II describes the models related to the Merise method based on which the database of the information system is structured, and consists of: conceptual data model, logical data model and physical model.

Chapter III details the description and functionality of the application, as well as the tools used to create the system. There are screenshots with the role of presenting in more detail how the IT system works.

Chapter IV includes the conclusions and suggestions for improving the system, as well as the bibliographic sources used.

Chapter 2. Analysis and design of the computer system

2.1 System analysis

The management and provision of services provided by hotels, in a short period of time and with minimal costs, represents an important role in maximizing profit. This can be achieved with the help of an IT system that includes various modules that allow internal activities to be carried out.

The computer system of a hotel is composed of several components that allow the fulfillment of certain social and economic objectives. In business, a hotel is a dynamic and complex organizational system.

Complexity is due to the number of components that, in turn, perform certain tasks. If complexity is seen from the perspective of functionality, then a hotel's IT system may include components such as reservation management, accommodation management, service delivery, human resource management, financial accounting, etc.

From the point of view of dynamics, the system involves numerous probabilities regarding the management of activities. It must be able to handle several types of operations depending on certain situations.

The main activities of a hotel are the booking of rooms and the accommodation of customers in free rooms, activities that are managed by the receptionist.

Booking the rooms implies the possibility to ensure the reserved number of rooms for the desired room category at the accommodation, so that the customer can choose any free room he wants.

Customers can book one or more rooms of the same type or of different types. They must provide the receptionist with information such as: the start date of the stay, the end date and the number of companions. After providing this information, the receptionist will check through the application if, for the desired period, there are rooms available with enough seats for all guests. After determining room availability, the receptionist will select the number and types of rooms the customer wishes to reserve.

After selecting the desired rooms, the client will be asked if he wishes to benefit from the paid services provided by the hotel. The service fee will be calculated according to the number of nights and the number of companions. In order to confirm the accommodation, the receptionist will request the customer's personal data to be registered in the database.

There are two types of accommodation: with reservation and without reservation. For accommodation without a reservation, the receptionist will ask the customer how many nights he wishes to stay at the hotel and if he has companions. As in the booking process, the application will determine the available rooms according to the requested information, the difference being that for accommodation, the remaining free rooms in which the customer wishes to stay will be selected, while for reservations, the available room categories will be selected. Next, there will be the possibility to select services, following the confirmation of the accommodation after the client is registered in the database.

Accommodation with reservation requires the provision of the CNP for clients who are individuals, or the Unique Registration Code for legal entities. Through the application, the receptionist will be able to see the list of the client's reservations. After selecting a reservation, the accommodation form will be automatically filled in with the selected services and the customer's personal data, after which he will choose the desired free rooms according to the number of rooms related to the reserved room categories.

Hotel administrators are employees who have access through the application to manage information about rooms, services, rates, discounts, employees and customers. They can periodically update room rates and discounts and generate graphs for specific periods to observe the evolution of the hotel's goals.

2.2 Designing the IT system

In this work, the Merise method, developed by the Technical Informatics Center of the French Ministry of Industry, during 1978-1979, was used as a methodology for the design and development of the IT system.

The Merise method consists in making models at the conceptual, organizational and operational level, thus realizing the existing links in the information systems.

The conceptual level represents the analysis of the information system from the point of view of the objectives. This involves the conceptual study of data, their organization and processing.

The organizational level involves the integration in the analysis of the criteria related to the organization and expresses the reality perceived as a whole regardless of the participating actors. From a data perspective, it is necessary to focus on a class of solutions, where the logical data model (MLD) appears and represents a textual transcription of the conceptual data model.

The operational level represents the means that will be effectively used to manage data and processing, including the provision of technical solutions. The physical data model (MFD) is built at this level and represents the transformation of the logical data model by equating the concepts of tables and columns with the concepts of relationships and attributes in the logical data model.¹

2.2.1 Conceptual data model

The Conceptual Data Model (CDM) allows the representation of data by means of appropriate graphics that reflect the static description of the information system, using entity and relationship concepts.

Entities are representations of physical or immaterial objects in an information system according to data management needs. Relationships between entities are represented by associations.

As part of this work, the conceptual model of the data relating to the IT system was designed, presented in *Figure 2.2.1.1* .

¹ Georgescu, C., *The relational and object approach in the analysis of computer systems* , RA Bucharest Didactic and Pedagogical Publishing House, 2002, page 80

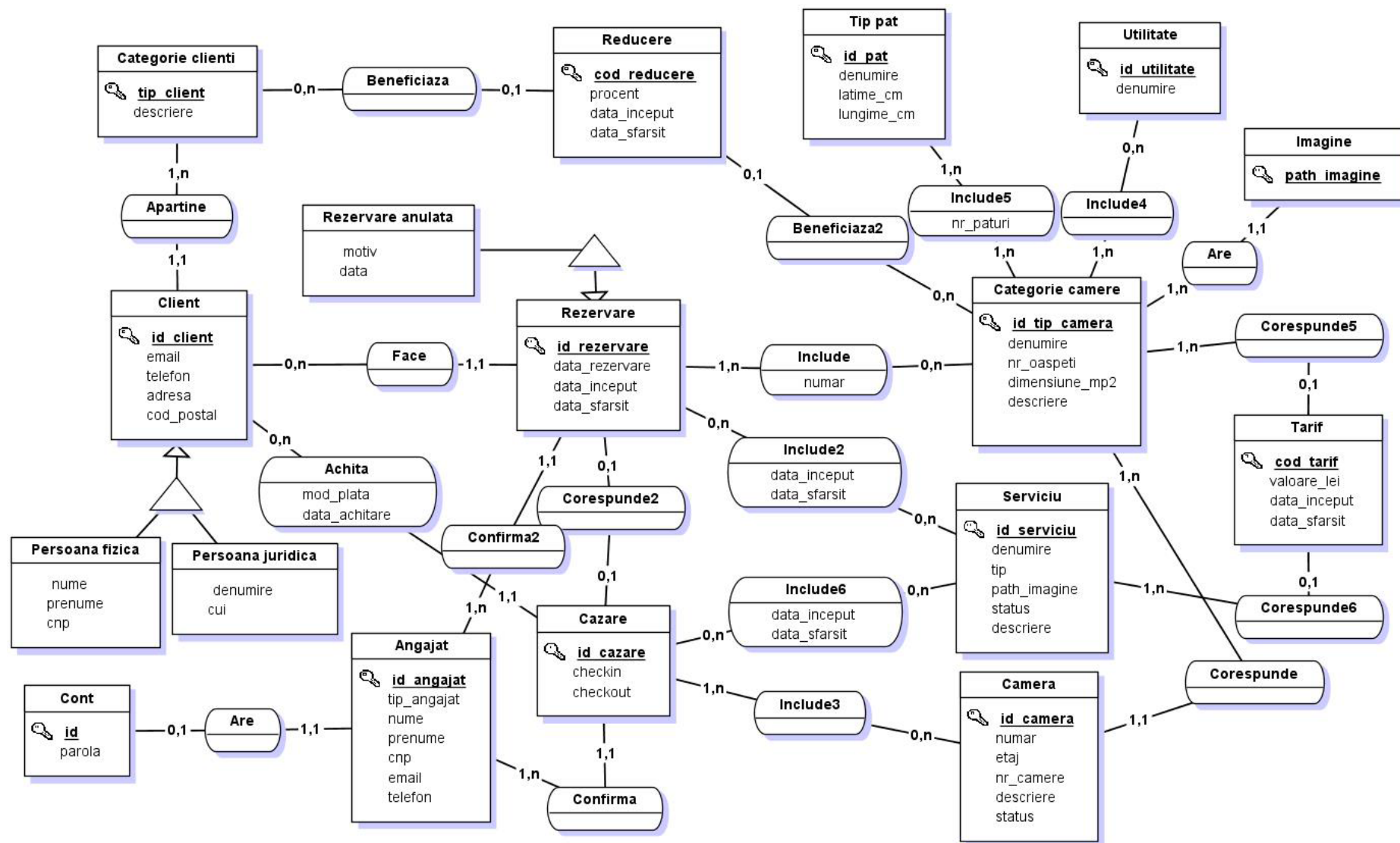


Figure 2.2.1. 1Conceptual data model

Next, the entities in the conceptual data model will be presented. Some entities have been designed to allow data to be retained for history management.

Client entity contains data about registered natural and legal persons. It is identified by the *id_client* attribute and is inherited by the entities *Natural Person* and *Legal Person* which include specific data such as *name* , *CUI code* as well as *last name* , *first name* and *CNP* . This favors the ability to manage discounts according to the type of customers.

Customer Category entity reflects the groups to which customers belong, thus it is possible to allocate discounts according to certain customer categories.

Between *the Customer* and *Customer Category entities* there is a 1,1 and 1,n relationship: any customer belongs to a group, and a group can contain one or more customers.

Discount entity is identified by the *discount_code* attribute and manages the discount history. The *discount_code* identifier stores a code of the form "R_CM1_2", where "T" - rate, "CM" - room category, "1" - room category id and "2" - the number of the discount related to the category with the id "1" . For the codes related to the discounts offered to customer categories, the code will have the form "R_CL1_2", the difference being the notation "CL" and means the customer category along with its id.

The relationship between *Customer Category* and *Discount* has cardinality 0,n and 0,1: any customer category can benefit from none or more discounts, and any discount can benefit from none or one customer category. Through this link, it will be possible to query the history of discounts assigned to customer categories.

Room Category entity contains information about the room categories within the hotel. It is identified by the *id_tip_camera* attribute and not by the name because this allows the registration in the database of several categories with the same name but with different information, in the situation where an update of the information is needed but at the same time the data in historic.

Also between *Category entities room* and *Discount* have cardinality 0,n and 0,1: any room category can benefit from none or more discounts, and one discount can benefit from none or one room category.

Utility entity includes the names of utilities found in rooms that are part of a given room category. For example, one room may include air conditioning, internet, safe, etc., but other rooms may have different utilities. Thus it is possible to manage these utilities.

Between *Utility* and *Room Category* there is the connection with cardinality 0,n and 1,n: any utility may be included in none or more room categories, and any room category may include one or more utilities.

Room entity contains data about hotel rooms. As with the *Camera Category entity* , this entity is identified by the *camera_id attribute* , and its number to keep historical data unchanged.

Between the entities *Category rooms* and *Rooms* there is cardinality 1,n and 1,1; any room category can have one or more rooms corresponding to it, and a room can correspond to at most one room category.

Service entity manages information about the services provided by the hotel. The *path_image* attribute contains the address where the representative image is saved on the server.

Accommodation entity manages the accommodation registered in the application. They are identified by *accommodation_id* and *the checkin* and *checkout attributes* will store the start and end date of the accommodation, based on which the room availability algorithm will work.

status attribute within *the Room* and *Service entities* reflects their availability. If any of these will not be available for a certain period of time, in order not to appear available in the application, the status will change from "active" to "inactive".

Reservation entity manages information about customer reservations. Between the *Accommodation* and *Booking entities* there is cardinality 0,1 and 0,1; any accommodation can correspond to none or a maximum of one reservation, and a reservation can correspond to none (if it is canceled before the accommodation) or a maximum of one accommodation.

Canceled Booking entity inherits from *the Booking entity* and manages all canceled bookings, reason and date. There is an inheritance relationship between *Canceled Booking* and *Booking* .

The link between *Customer* and *Reservation* has cardinality 0,n and 1,1; any customer can make none or more reservations, and one reservation can be made by one customer.

The *Employee* entity manages the data of employees using the software application. The *Account* entity contains the login credentials of the employees to be able to access the software application. The *password* attribute will contain an encoded text to increase data security.

Employee and *Account* entities are related with cardinality 1,1 and 1,1: every employee has an account, and an account belongs to an employee.

Employee entity is linked by two *Confirm* associations to the *Accommodation* and *Reservation* entities, both having cardinality 1,n and 1,1: any employee can confirm one or more accommodations / reservations, and a reservation / accommodation can be confirmed by an employee.

Between *the Client* and *the Accommodation* there is the *paid* relationship which contains information about the date the clients stayed and paid, as well as the method of payment used.

The association between *Reservation* and *Service* and the association between *Accommodation* and *Service* have cardinality 0,n and 0,n: any reservation or accommodation can include none or more services, and a service can be included in none or more reservations or accommodations. Thus, accommodations can have associated services regardless of whether a reservation has been made or not.

BedType entity contains information about the types of beds available in the hotel. *the Room Category* and *Bed Type* entities has cardinality 1,n and 1,n: any room category can have one or more beds included, and a bed can be included in one or more room categories. In this link, the *nr_beds* attribute appears, which will specify how many beds from a certain bed category will be assigned to the room categories.

the Booking and *RoomCategory* entities has cardinality 1,n and 0,n: any booking can include more than one room category, and a room category can be included in none or more bookings. Thus, the *number* attribute appears in the association that specifies how many rooms from the selected room category the customer wants.

The association between *Accommodation* and *Room* has cardinality 1,n and 0,n: any accommodation can include one or more rooms, and a room can be included in none or more accommodations.

Rate entity contains data about room rates within room categories and services. This entity is identified by the *tariff_code* attribute that will differentiate by entity. For example, for the tariff of a category, the code will have the structure: "T_C1_2", where: "T" - tariff, "C" - category, "1" - the category id, "2" - the number of the tariff related to the category with id "1". For the tariff of a service, the code will have the structure: "T_S1_1", the difference being the character "S" which represents the service.

The rate has a daily value in lei, which will be calculated according to the service or room chosen, and the duration of days selected.

The link between *Rate* and *Room Category* , as well as that between *Rate* and *Service* , have cardinality 0.1 and 1.n: any tariff corresponds to none or to a category of rooms / service, and to a category of rooms / a service corresponds to one or more tariffs.

2.2.2 Logical data model

The transition from the conceptual data model to the logical data model considers two main types of rules, which are related to the distribution of data across entity types and the cardinality of associations. These methodological elements are called transformation rules.

Rule 1. Existing entities become tables, so properties become table columns and entity identifiers become primary keys.

Rule 2. When in a binary link there are cardinalities 0.1 or 1.1 and 0.n or 1.n, then the primary key from the entity with cardinality 0.n or 1.n migrates to the other entity.

Rule 3. A binary relation that has links of cardinality 0,n or 1,n will become a table in the logical data model, where the primary keys of the entities will migrate to the new table. Thus, the primary key of the table-transformed association will be composed of the primary keys of the entities. If the relation contains properties, they will become columns in the resulting table.

Rule 4. If a binary relation has two links of cardinality 0.1 or 1.1 then a double migration of entity identifiers occurs, resulting in them appearing as foreign keys in each entity.

Rule 5. In the case of parent and specific entities, a table will be created for each entity by migrating the identifier and properties from the parent entity to the child entity.

According to the previously mentioned rules, the logical model of the system data is created, presented in *Figure 2.2.2.1* .

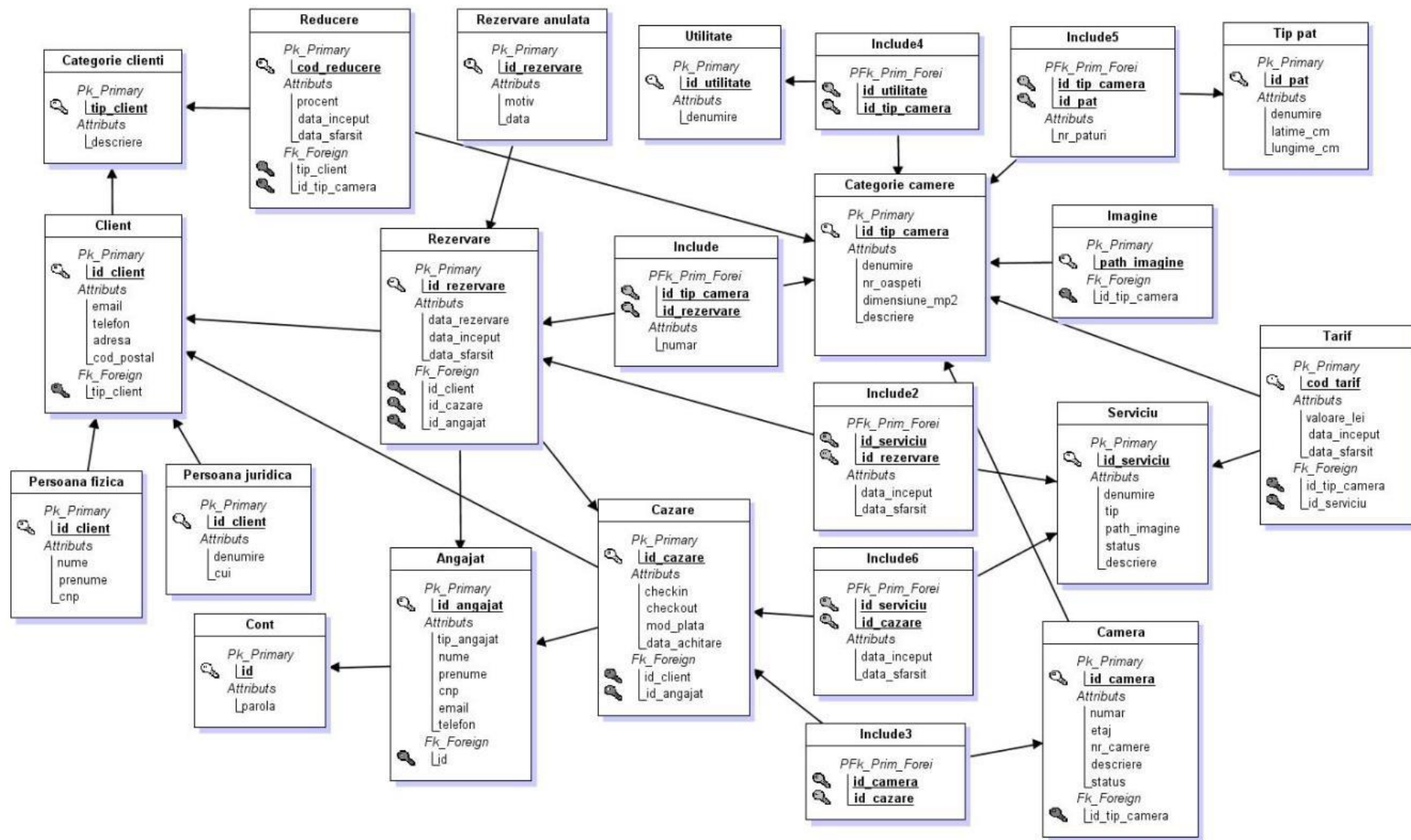


Figure 2.2.1. 2The logical data model

Information about the resulting logical data model according to the transformation rules of the conceptual data model will be presented.

According to the rules of moving to the logical data model, entities become tables and properties become table columns.

The relationship between *Customer* and *Customer Category* , having cardinality 1,1 and 1,n, resulted in the migration of the primary key from *Customer Category* to the *Customer* entity as a foreign key.

The inheritance relationships between the parent entity *Customer* and the entities *Person* and *Entity* resulted in the migration of the *customer_id* attribute to the child entities as the primary key.

Canceled Booking entity that inherited *Booking* in the conceptual data model, according to the conversion rules to the logical model, the *reservation_id* attribute migrated from the parent entity to the child entity.

Between the *Customer* , *Employee* , *Accommodation* , and *Reservation* entities, the primary keys *client_id* , *accommodation_id* , and *employee_id* were migrated to the *Reservation* entity as foreign keys. Thus, for each reservation, information can be found about the client who requested the reservation, the accommodation related to the reservation, as well as information about the employee who confirmed the reservation.

The cardinality 1,n and 0,n of the association between the *Booking* and *RoomCategory* entities migrated both primary keys to the association, becoming a new primary key table consisting of the migrated primary keys, along with the column *number* . Thus, this table will contain data about the number of rooms booked from a certain room category.

The associations between the entities *Booking - Service* and *Accommodation - Service* were transformed into tables due to the cardinalities 0,n and 0,n, so that the primary keys from the entities migrated to the tables forming one primary key composed of both migrated columns. These resulting tables will contain the period of services selected when booking and accommodation.

the *Client* and *Accommodation* entities with cardinality 0,n and 1,1 resulted in the migration of the primary key *client_id* as a foreign key, and the attributes in the association as columns, to the *Accommodation* entity . It also migrates the primary key from the *Employee* entity as a foreign key, thus resulting in a table that will include customer, employee, check-in date, and payment method data.

The association between the entities *Accommodation* and *Room* with cardinality 1,n and 0,na led to the migration of the primary keys from both tables to the table, forming the composite primary key.

Between the *Employee* and *Account* entities there is the relationship with cardinality 1,1 and 1,1, which resulted in migrating the primary key *id* from *Account* to *Employee* as a foreign key.

The link between *Camera Category* and *Camera* with cardinality 1,n and 1,1 resulted in migrating the primary key *id_type_camera* to *Camera* as a foreign key.

Between the entities *Room Category – Tariff* and *Service – Tariff* there are links with cardinality 1,n and 0,1, which resulted in the migration of primary keys *id_type_room* and *id_room* to *Tariff* as foreign keys.

The cardinality 1,n and 1,n related to the link between the entities *Room Category* and *Bed Type* led to the migration of the primary keys from both entities to the created table, forming the primary key composed of both columns. This table also contains the *nr_beds column* , thus managing information about the number of beds and their type found in the room categories.

Between *Category rooms* and *Utility* there is the connection with the cardinality 1,n and 0,n which led to the appearance of a table with the primary key composed of the primary keys of the entities. In this table manage the utilities that a room category has.

The link between *Camera Category* and *Image* with cardinality 1,n and 1,1 resulted in the migration of the primary key *id_type_camera* to the *Image table* , becoming a foreign key. Thus, the table will manage the images related to the camera categories.

2.2.3 The physical model of the data

The physical data model involves the transformation of the logical data model related to a database management system.

The Microsoft SQL database server was used in the developed application. The Database Management System is a software product that stores and retrieves data requested by other applications and represents all the programs used to create, query and maintain a database.

Next, each table in the database used in this work will be detailed.

	Name	Data Type	Allow Nulls	Default
PK	id_client	int	<input type="checkbox"/>	
	email	nvarchar(50)	<input checked="" type="checkbox"/>	
	telefon	nvarchar(50)	<input checked="" type="checkbox"/>	
	adresa	nvarchar(100)	<input type="checkbox"/>	
	cod_postal	nvarchar(50)	<input type="checkbox"/>	
			<input type="checkbox"/>	

Figure 2.2.3. 1Client table

Figure 2.2.3.1 shows the structure of the *client table* , a table that manages the information of clients registered after booking or staying. The primary key of the table is the *id_client* int type field , next to which the *Allow Nulls option is unchecked* because this field cannot accept *NULL values* . The rest of the fields are of character type, *nvarchar* of size 50 characters, except the *address field* with 100 characters.

	Name	Data Type	Allow Nulls	Default
PK	id_client	int	<input type="checkbox"/>	
	nume	nvarchar(50)	<input type="checkbox"/>	
	prenume	nvarchar(50)	<input type="checkbox"/>	
	cnp	nvarchar(50)	<input type="checkbox"/>	
			<input type="checkbox"/>	

Figure 2.2.3. 2Table of physical_persons

Figure 2.2.3.2 shows the structure of the *physical_persons table* , with the primary key *id_client* of type int and fields of character type, *nvarchar* with the size of 50 characters. This table will manage the information of natural persons registered in the application.

dbo.persoane_juridice [Design] -> X				
Update	Script File: dbo.persoane_juridice.sql			
	Name	Data Type	Allow Nulls	Default
	id_client	int	<input type="checkbox"/>	
	denumire	nvarchar(50)	<input type="checkbox"/>	
	cui	nvarchar(50)	<input type="checkbox"/>	
			<input type="checkbox"/>	

Figure 2.2.3. 3Table of legal_persons

The table in *figure 2.2.3.3* shows the structure of the *legal_entities* table , with the primary key *id_client* of type *int* and fields of character type, *nvarchar* with the size of 50 characters. This table manages information about legal entities registered in the application.

dbo.categorii_clienti [Design] -> X				
Update	Script File: dbo.categorii_clienti.sql			
	Name	Data Type	Allow Nulls	Default
	tip_client	nvarchar(50)	<input type="checkbox"/>	
	descriere	nvarchar(50)	<input checked="" type="checkbox"/>	
			<input type="checkbox"/>	

Figure 2.2.3. 4Client_categories table

The table in *figure 2.2.3.4* illustrates the structure of the *customer_categories* table, having *nvarchar* type fields with a size of 50 characters, and the primary key named *customer_type* . This table manages the categories of customers who will be able to benefit from discounts.

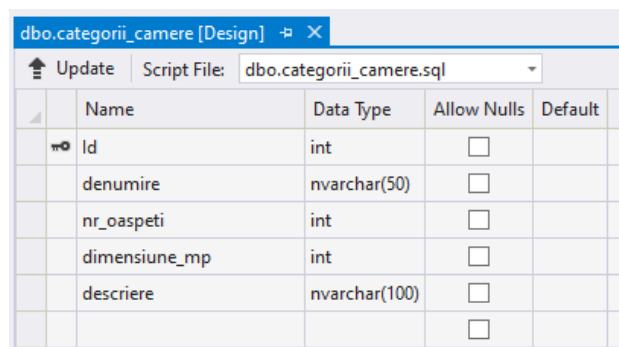
dbo.reduceri [Design] -> X				
Update	Script File: dbo.reduceri.sql			
	Name	Data Type	Allow Nulls	Default
	cod_reducere	nvarchar(50)	<input type="checkbox"/>	
	procent	int	<input type="checkbox"/>	
	data_inceput	datetime	<input type="checkbox"/>	
	data_sfarsit	datetime	<input type="checkbox"/>	
	id_categorie_camera	int	<input type="checkbox"/>	
	tip_client	nvarchar(50)	<input type="checkbox"/>	
			<input type="checkbox"/>	

Figure 2.2.3. 5Table of discounts

Figure 2.2.3.5 shows the structure of the *discounts* table that manages the discounts within the hotel. *Cod_reducere* is the primary key of *nvarchar* type with the size of 50 characters. Found foreign keys *camera_id_category* and *client_type* , fields that cannot contain data or *NULL* at the same time because a field must always be *NULL* . Thus, it will be possible to distinguish the entity to which the discount will be applied (for example, if the *client_type* field is *NULL* and the *room_category_id* contains a value, it means that the discount is applied to a room category).

percentage field expresses the percentage by which the price of a room category is reduced. For example, a student with a 10% discount (discount applied per customer category) staying in a single room with a 10% discount (discount applied per room category) will pay 80% of the total price , because these two types of discounts are cumulative.

start_date and *end_date* fields express the period during which the discount is valid. They will store *datetime* data .



The screenshot shows the 'dbo.categorii_camere [Design]' window in SQL Server Enterprise Designer. The 'Script File' dropdown is set to 'dbo.categorii_camere.sql'. The table structure is as follows:

	Name	Data Type	Allow Nulls	Default
PK	Id	int	<input type="checkbox"/>	
	denumire	nvarchar(50)	<input type="checkbox"/>	
	nr_oaspeti	int	<input type="checkbox"/>	
	dimensiune_mp	int	<input type="checkbox"/>	
	descriere	nvarchar(100)	<input type="checkbox"/>	
			<input type="checkbox"/>	

Figure 2.2.3. 6Room_categories table

Figure 2.2.3.6 shows the structure of the *room_categories* table that manages the information related to the room categories. The primary key *Id* refers to the field *id_type_camera* from the models presented in this paper. The *no_guests* field expresses the recommended number of guests for a certain room category. *Size_square* stores the size in square meters of the rooms related to the category and the *description* column will memorize a short description.

	Name	Data Type	Allow Nulls	Default
	id_camera	int	<input type="checkbox"/>	
	numar	nvarchar(50)	<input type="checkbox"/>	
	id_categorie_camere	int	<input type="checkbox"/>	
	etaj	int	<input type="checkbox"/>	
	nr_camere	int	<input type="checkbox"/>	
	descriere	nvarchar(50)	<input type="checkbox"/>	
	status	nvarchar(50)	<input type="checkbox"/>	
			<input type="checkbox"/>	

Figure 2.2.3. 7Table of rooms

The table in *figure 2.2.3.7* shows the structure of the *room* table that contains the information of the rooms related to the room categories. The existence of the *id_camera* primary key instead of the *number* column allows the registration of several cameras with the same number in order to record a history of cameras that have changed over time, but without affecting the history data. The *status* column indicates the availability of the camera and stores one of the following values : " active" or "inactive". Cameras with the status " inactive" will not appear in the camera availability algorithms. This status is useful if the hotel decides that a room cannot be occupied for a certain period, until the status is changed back to " active".

	Name	Data Type	Allow Nulls	Default
	Id	int	<input type="checkbox"/>	
	denumire	nvarchar(50)	<input type="checkbox"/>	
			<input type="checkbox"/>	

Figure 2.2.3. 8Utility table

utilities table in *figure 2.2.3.8* contains data about the utilities of the rooms within the hotel.

dbo.utilitati_camere [Design] ↗ ✕					
Update		Script File: dbo.utilitati_camere.sql			
	Name	Data Type	Allow Nulls	Default	
PK	id_categorie_camere	int	<input type="checkbox"/>		
PK	id_utilitate	int	<input type="checkbox"/>		
			<input type="checkbox"/>		

Figure 2.2.3. 9Table utilities_rooms

room_utilities table shown in *figure 2.2.3.9* stores the utilities assigned to the room categories.

dbo.paturi [Design] ↗ ✕					
Update		Script File: dbo.paturi.sql			
	Name	Data Type	Allow Nulls	Default	
PK	id_tip_pat	int	<input type="checkbox"/>		
	denumire	nvarchar(50)	<input type="checkbox"/>		
	latime	int	<input checked="" type="checkbox"/>		
	lungime	int	<input checked="" type="checkbox"/>		
			<input type="checkbox"/>		

Figure 2.2.3. 10Table of beds

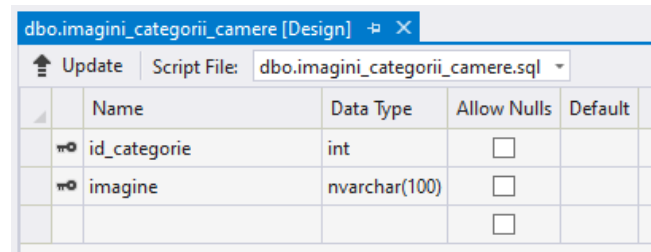
In *figure 2.2.3.10* the *beds* table is illustrated , which manages information about the existing bed types within the hotel. The int *width* and *length* fields store the dimensions in centimeters of the beds.

dbo.paturi_categorii_camere [Design] ↗ ✕					
Update		Script File: dbo.paturi_categorii_camere.sql			
	Name	Data Type	Allow Nulls	Default	
PK	id_categorie_camera	int	<input type="checkbox"/>		
PK	id_tip_pat	int	<input type="checkbox"/>		
	nr_paturi	int	<input type="checkbox"/>		
			<input type="checkbox"/>		

Figure 2.2.3. 11Table beds_categories_rooms

In *figure 2.2.3.11* the table *beds_categories_rooms* is presented with the primary key composed of the fields *id_category_room* and *id_bed_type* . The table manages the

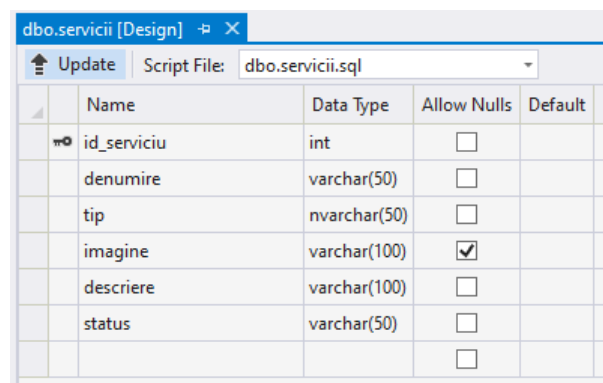
beds assigned to the room categories. The *no_beds* field indicates the number of beds assigned to a room, which can contain several beds of different types. For example, a "King Apartment" room category can have one "double bed" and two "single beds".



	Name	Data Type	Allow Nulls	Default
PK	id_categorie	int	<input type="checkbox"/>	
PK	image	nvarchar(100)	<input type="checkbox"/>	

Figure 2.2.3. 12Table *images_categories_rooms*

In *figure 2.2.3.12* the table *images_categories_rooms* is presented , having the primary key consisting of the fields *id_category* and *image* . The *image* field contains the path to the file containing the camera image. This table manages all the images that a camera category can have .



	Name	Data Type	Allow Nulls	Default
PK	id_serviciu	int	<input type="checkbox"/>	
	denumire	varchar(50)	<input type="checkbox"/>	
	tip	nvarchar(50)	<input type="checkbox"/>	
	image	varchar(100)	<input checked="" type="checkbox"/>	
	descriere	varchar(100)	<input type="checkbox"/>	
	status	varchar(50)	<input type="checkbox"/>	
			<input type="checkbox"/>	

Figure 2.2.3. 13Services table

services table in *figure 2.2.3.13* stores all the services provided by the hotel. Primary key of type *int* is represented by *the service_id field* . The *type* field can contain one of the following values : " occasional" or "daily". Occasional services refer to those services that can only be purchased for certain days of the stay. Daily services are available throughout the stay. Both types of services have a daily cost in lei.

image field stores the path to the representative image that will be displayed in the application.

As in the case of the *Camera table* , the status can have one of the following values : "active" or "inactive". Inactive services will not be able to be selected in the reservation and accommodation process, if the hotel cannot make them available for a period, until the status is changed back to "active".

Name	Data Type	Allow Nulls	Default
cod_tarif	nvarchar(50)	<input type="checkbox"/>	
valoare_lei	decimal(5,2)	<input type="checkbox"/>	
data_inceput	datetime	<input type="checkbox"/>	
data_sfarsit	datetime	<input type="checkbox"/>	
id_categorie_camera	int	<input checked="" type="checkbox"/>	
id_serviciu	int	<input checked="" type="checkbox"/>	
		<input type="checkbox"/>	

Figure 2.2.3. 14Tariff table

In figure 2.2.3.14 the *rate table* is presented in which the rates of room categories and services will be managed. The primary key is represented by the nvarchar code_tariff field with a size of 50 characters, which will store a coded text of the form T_C1_2 (T – “tariff”, C – “ room”, “1” – room category id , "2" – the tariff number related to the category) or T_S1_2, the difference being the character "S" which stands for service.

The decimal value_lei field can contain a number consisting of a maximum of 5 digits with 2 decimal places and stores the value in lei of the tariff for the period specified in the *start_date* and *end_date* fields . Thus, when querying the table, the history of tariffs can be found according to the period. The decimal type was chosen instead of double because of the higher precision

The *room_category_id* and *service_id* fields are primary keys of the *room_category* and *services* tables and foreign keys in this table. One of these fields will be NULL because the same rate cannot apply to both entities.

Name	Data Type	Allow Nulls	Default
id	nvarchar(50)	<input type="checkbox"/>	
parola	nvarchar(50)	<input type="checkbox"/>	
		<input type="checkbox"/>	

Figure 2.2.3. 15Table of accounts

account table shown in *figure 2.2.3.15* stores the login data of the employees in the software application. The primary key is represented by the *id* field and will be entered manually. *Password* field it will contain an encrypted password. Both columns are of type *varchar*.

	Name	Data Type	Allow Nulls	Default
PK	id_angajat	int	<input type="checkbox"/>	
	tip_angajat	nvarchar(50)	<input type="checkbox"/>	
	nume	nvarchar(50)	<input type="checkbox"/>	
	prenume	nvarchar(50)	<input type="checkbox"/>	
	cnp	nvarchar(50)	<input type="checkbox"/>	
	email	nvarchar(50)	<input type="checkbox"/>	
	telefon	nvarchar(50)	<input type="checkbox"/>	
	id_cont	nvarchar(50)	<input type="checkbox"/>	

Figure 2.2.3. 16Table of employees

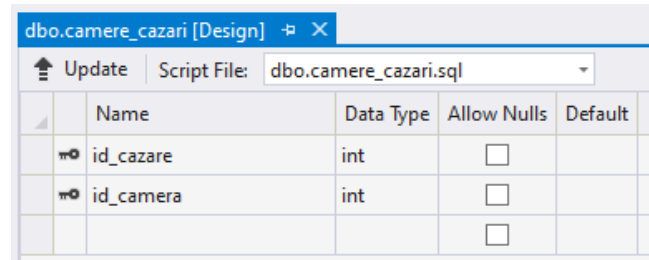
In *figure 2.2.3.16* illustrated is the structure of the *employees* table that will store the data of the employees using the software application. The "employee_type" column will store one of the following values: "receptionist" or "administrator". Depending on this value, the employee will have access to different forms within the application.

	Name	Data Type	Allow Nulls	Default
PK	id_cazare	int	<input type="checkbox"/>	
	id_client	int	<input type="checkbox"/>	
	checkin	datetime	<input type="checkbox"/>	
	checkout	datetime	<input type="checkbox"/>	
	mod_plata	nvarchar(50)	<input type="checkbox"/>	
	total_achitat_lei	decimal(10,2)	<input type="checkbox"/>	
	id_angajat	int	<input type="checkbox"/>	

Figure 2.2.3. 17Accommodation table

In the *figure 2.2.3.17* shows the structure of the *accommodation* table that manages the accommodation registered within the hotel. The *total_paid_lei* field stores the total amount in lei of all the rooms and services selected by the customer and is a *decimal* type

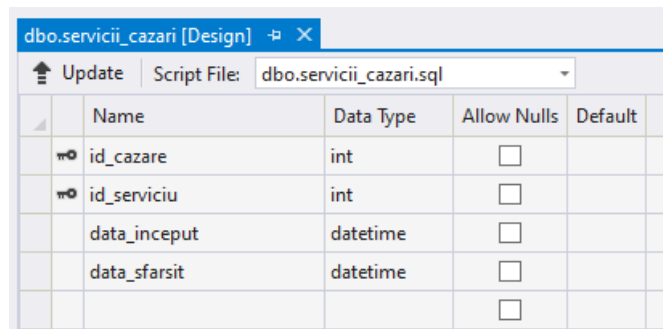
of 10 characters with 2 decimal places due to the high precision of the calculations. *Mod_plata* stores the payment method used ("card" or "cash") and the *checkin* and *checkout* fields indicate the period of the stay (start date, respectively end date of the stay).



	Name	Data Type	Allow Nulls	Default
PK	id_cazare	int	<input type="checkbox"/>	
PK	id_camera	int	<input type="checkbox"/>	
			<input type="checkbox"/>	

Figure 2.2.3. 18Table of rooms_accommodations

room_accommodation table shown in *figure 2.2.3.18* will manage the record of hotel rooms allocated to accommodation. The primary key is composed of two fields due to the cardinality 1,n, thus being able to register several rooms at one accommodation.



	Name	Data Type	Allow Nulls	Default
PK	id_cazare	int	<input type="checkbox"/>	
PK	id_serviciu	int	<input type="checkbox"/>	
	data_inceput	datetime	<input type="checkbox"/>	
	data_sfarsit	datetime	<input type="checkbox"/>	
			<input type="checkbox"/>	

Figure 2.2.3. 19Services_accommodation table

Figure 2.2.3.19 shows the *services_accommodations* table that manages the services assigned to an accommodation. The *date_start* and *date_end* fields of type *datetime* store the duration during which a particular service is valid for accommodation.

	Name	Data Type	Allow Nulls	Default
	id_rezervare	int	<input type="checkbox"/>	
	id_client	int	<input type="checkbox"/>	
	id_cazare	int	<input checked="" type="checkbox"/>	
	data_rezervare	datetime	<input type="checkbox"/>	
	data_inceput	datetime	<input type="checkbox"/>	
	data_sfarsit	datetime	<input type="checkbox"/>	

Figure 2.2.3. 20Reservation table

Figure 2.2.3.20 shows the structure of the *reservation table* that will manage data about the reservations made within the hotel. The fields *client_id* and *accommodation_id* of type *int* are foreign keys of the related tables, where *accommodation_id* can be *NULL* due to the fact that some reservations can be canceled in time, without the actual accommodation of the client taking place. The *start_date* and *end_date* columns indicate the period of stay desired by the client. The *reservation_date* field stores the date the reservation took place.

	Name	Data Type	Allow Nulls	Default
	id_tip_camera	int	<input type="checkbox"/>	
	id_rezervare	int	<input type="checkbox"/>	
	numar_camere	int	<input type="checkbox"/>	

Figure 2.2.3. 21Table of reserved_rooms

Figure 2.2.3.21 shows the *reserved_rooms table* that manages the categories of rooms assigned to a reservation. For example, a customer can book two "Single room" rooms and one "Double room" room. Thus, the structure of this table allows this, thanks to the *number_rooms* field that indicates the number of rooms in the desired room category.

	Name	Data Type	Allow Nulls	Default
PK	id_rezervare	int	<input type="checkbox"/>	
PK	id_serviciu	int	<input type="checkbox"/>	
	data_inceput	datetime	<input type="checkbox"/>	
	data_sfarsit	datetime	<input type="checkbox"/>	
			<input type="checkbox"/>	

Figure 2.2.3. 22Services_reservations table

Similar to the structure of the *services_accommodations* table shown in figure 2.2.3.19 , the *services_reservations* table fulfills the same role of managing the services assigned to a reservation, as well as the period in which they will be available thanks to the *start_date* and *end_date* fields that indicate the duration of the service. This table is necessary because in the process of accommodation the client can select both the desired rooms and the services. At the time of accommodation, the data from this table will be saved in the *services_accommodation* table .

Chapter 3. Description of the computer system

In this chapter, the computer system developed in this work, which has the main purpose of booking rooms and services, and accommodating customers, will be presented in detail . Screenshots will be presented to more explicitly present the functionality of the application.

The software application is intended for *the Administrator* and *Receptionist* components .

For the development of the software application we used *Windows Forms* , representing a set of libraries included in the .NET Framework (the platform for developing software applications, containing predefined classes and functions) developed by Microsoft.

The database was developed using *Microsoft SQL Server Management* .

Thus, in this work, an IT system will be proposed for the management of the activity at the level of a hotel. The necessary data will be stored in a database, and the computer system that processes and manages them will be composed of two main components, Administrator and Receptionist.

The Receptionist component has the role of managing both the reservations of the rooms and services offered by the hotel, for a determined period of time, as well as the accommodations. By means of this component, it will be possible to manage and realize the accommodation of customers at the hotel according to the available rooms, the desired services and the chosen period. The accommodation will be possible regardless of whether the customers have a reservation or not.

At the same time, there will be the option to reserve rooms and services according to the specified period, if customers want to make a reservation for a certain range of days.

The algorithm behind this process will make a series of checks to determine if there are rooms available, based on current accommodations and bookings. To complete the accommodation process, it will be checked if there is a reservation, if the customer is already registered in the database and if there are discounts for the category to which the customer belongs or seasonal discounts.

The Administrator component has the role of managing information about rooms, room categories, services, rates, discounts and reports generated according to the desired period.

The computer system is designed to allow the viewing of the history of accommodations, reservations, rates and discounts and at the same time to keep the data valid in case of updates.

3.1 Administrator component

Administrator component related to the IT system represents the "actor" who has access to the management of specific information within the hotel.

The applications were developed in the *Visual Studio 2013 programming environment*, an integrated program from *Microsoft* that allows the design of applications using the *.NET architecture*.

The programming language used to build the applications is C#, which is an object-oriented programming language that allows developers to build many types of robust and secure applications using the *.NET architecture libraries*.

.NET architecture defines the programming environment for developing and running applications.

Visual Studio provides tools that make it possible to develop applications in a safe and easy way. The most important are:

- the visual *designer interface* with drag-and-drop options that allow the implementation of classes in a dynamic way
- compilation and debugging
- code editor with syntax checker;

Windows applications made in C# typically display one or more windows containing controls that the user interacts with.

Next, the *Administrator component* of the software application will be presented in detail.

When starting the application, an authentication form shown in *figure 3.1.1 opens* in which the authors (*Administrator* and *Receptionist*) must enter the user ID and password to be able to access the forms related to the authors.

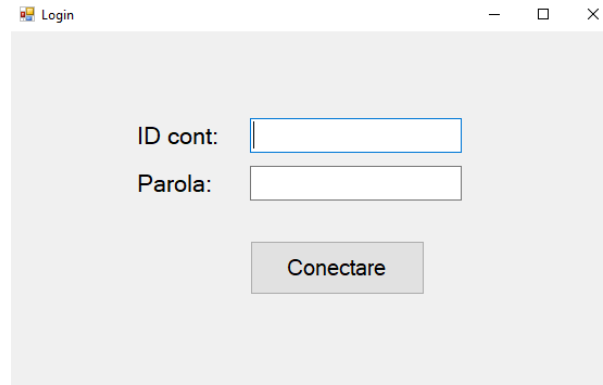


Figure 3.1. 1Login window

If the entered data is not correct, a warning message will be displayed (*figure 3.1.2*).

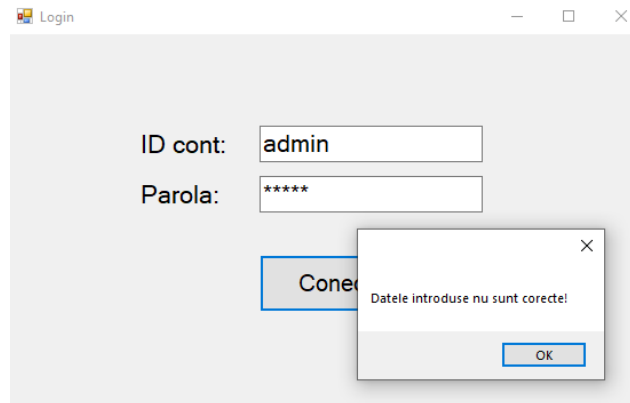


Figure 3.1. 2Login with wrong data

If the data entered is correct then a menu specific to the actor will open. In this case, the menu intended *for Administrators* will open (*figure 3.1.3*).

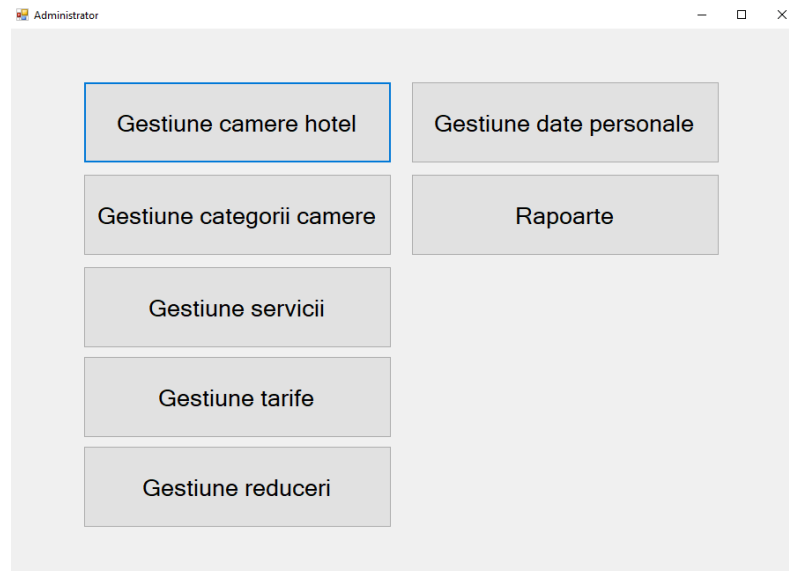


Figure 3.1. 3Administrators menu

In the menu shown in *figure 3.1.3.* there are several modules that can only be accessed by registered administrators (*Management of hotel rooms, Management of room categories, Management of services, Management of rates, Management of discounts , Management of personal data, Management of accounts, Reports*).

Upon pressing any button, the current window will close and a new window specific to the selected module will appear.

Hotel Rooms Management module , presented in *figure 3.1.4* , includes an interface that allows the user to view the rooms registered within the hotel, through a *DataGridView* type control in the form of a table.

Gestione camere

Numar camera: Cauta Reseteaza rezultate

Camera	Categorie	Etaj	Nr. camere	Descriere	Status
10	Camera single	1	1	Camera ori...	activ
11	Camera double	1	1	Camera ori...	activ
12	Camera single	1	1	Camera ori...	activ
13	Camera single	1	1	Camera ori...	activ
14	Camera single	1	1	Camera ori...	activ
15	Camera double	1	1	Camera ori...	activ

Detalii camera

Numar camera:

Categorie:

Status camera: ☐ inactiv

Etaj:

Numar camere:

Descriere:

Adauga camera Actualizeaza camera

Figure 3.1. 4The rooms management window

In the table you can find the information related to the rooms (room number, room category, floor, number of rooms, description and status). At the top there is a menu that allows you to filter the results. After entering a room number, pressing the *Search button* will update the list of rooms and return only the entered room, if any. The *Reset results* button will show back the original results.

On the right, the controls grouped in the *Camera Details section* allow entering new camera data and registering them in the database. After registration, the table will update and appear alongside the other rooms.

The status of the rooms indicates their availability. *Checkbox* control *camera status* indicates the status of a camera. If it is *inactive*, it will not appear in the accommodation form and cannot be selected. This option is valid in case the hotel can no longer have certain rooms for an indefinite period.

Gestiune categorii camere

Categorie	Limita oaspeti	Dimensiune	Descriere
Camera single	1	20	Camera pentru o ...
Camera double	2	30	Camera pentru do...
Apartament	4	100	Apartament pentr...
Camera twin	2	40	Camera twin cu d...

Descriere Paturi Utilitati Imagini

Denumire:
Camera single

Numar locuri:
1

Dimensiune
20

Descriere:
Camera pentru o singura persoana.

Adauga categorie Actualizeaza categorie

Figure 3.1. 5The room category management window

By means of the *room category management module* shown in *figure 3.1.5* , the data about the existing room categories in the hotel is managed. The table on the left shows the existing categories, and on the right there is a control with several tabs, each with different functions.

When selecting a category from the table, the data related to the selected category will be automatically filled in on the right side in all the tabs, so that the user can change the data in the easiest way. In the *Description tab* , pressing the *Update category button* will update the data of the selected category, and the *Add category button* will register a new category according to the entered data. There is the restriction that a newly introduced category does not have an already existing name, to prevent possible errors when trying to register in the database.

Figure 3.1. 6The " Beds" tab related to the selected category

In *figure 3.1.6* you can see the form in the *Beds tab* through which beds can be added or deleted for the selected category. The type of bed can be selected from a *ComboBox* type control , and the number of *beds* will be entered in Number of beds. The structure of the physical model of the database allows the allocation of several different beds to a room category.

If a category has a certain type of bed allocated, and it is desired to add an additional bed of the same type, it will be selected from *the ComboBox* and the number of beds will be entered. When the *Add bed button is pressed* , a warning will be displayed informing the user that the number of beds related to the selected bed type will be updated, as shown in *figure 3.1.7* .

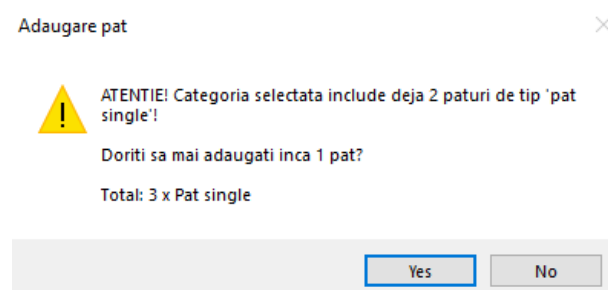


Figure 3.1. 7Warning message when adding an extra bed

Delete bed button will delete all selected beds in the list above.

Figure 3.1. 8" Utilities" tab related to the selected category

Utilities tab, you can find the list of utilities included in a room category. For example, there may be rooms that do not have air conditioning and others that do. As in the *Beds* tab , when selecting existing utilities from the list above, they can be deleted by pressing the *Delete utility button* . The *Add utility* button will assign the utility selected in the *combobox* to the selected category .

Figure 3.1. 9" Images" tab related to the selected category

Figure 3.1.9 shows the last tab related to the *Camera category management module* , called *Images* , where you can find the *Open gallery button* that will open a new window where you can set the image gallery related to the selected camera category.

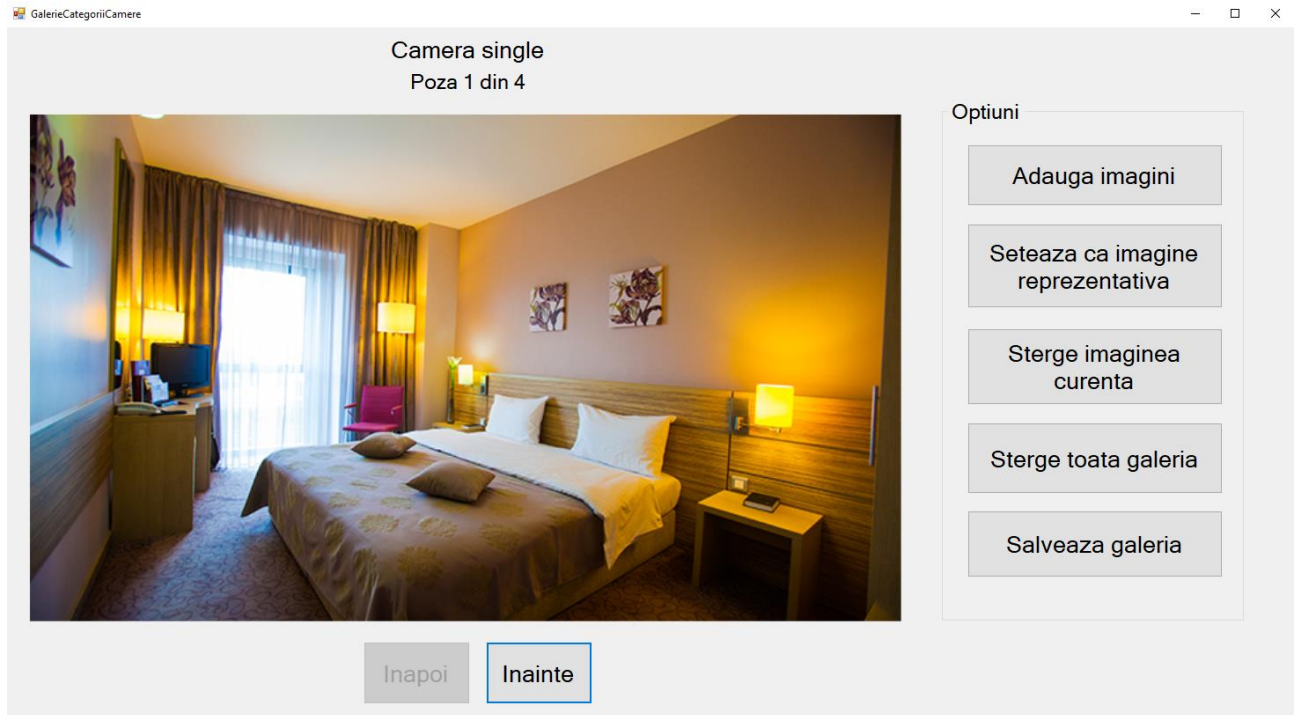


Figure 3.1. 10Image gallery of the selected camera category

In the window opened after pressing the *Open gallery button* , shown in *figure 3.1.10* , the images related to a category of cameras can be managed. At the bottom there are two buttons named *Back* and *Forward* with the role of navigating through the gallery. At the top, the selected camera category and the position of the image in the gallery ("Picture 1 of 4") are indicated.

On the right side there are several options to manage the gallery. The *Add images* button will open a window through which the administrator will be able to select images of type .jpg, .jpeg or .png, saved in the computer, and which will be added to the current gallery, as shown in *figure 3.1.11*.

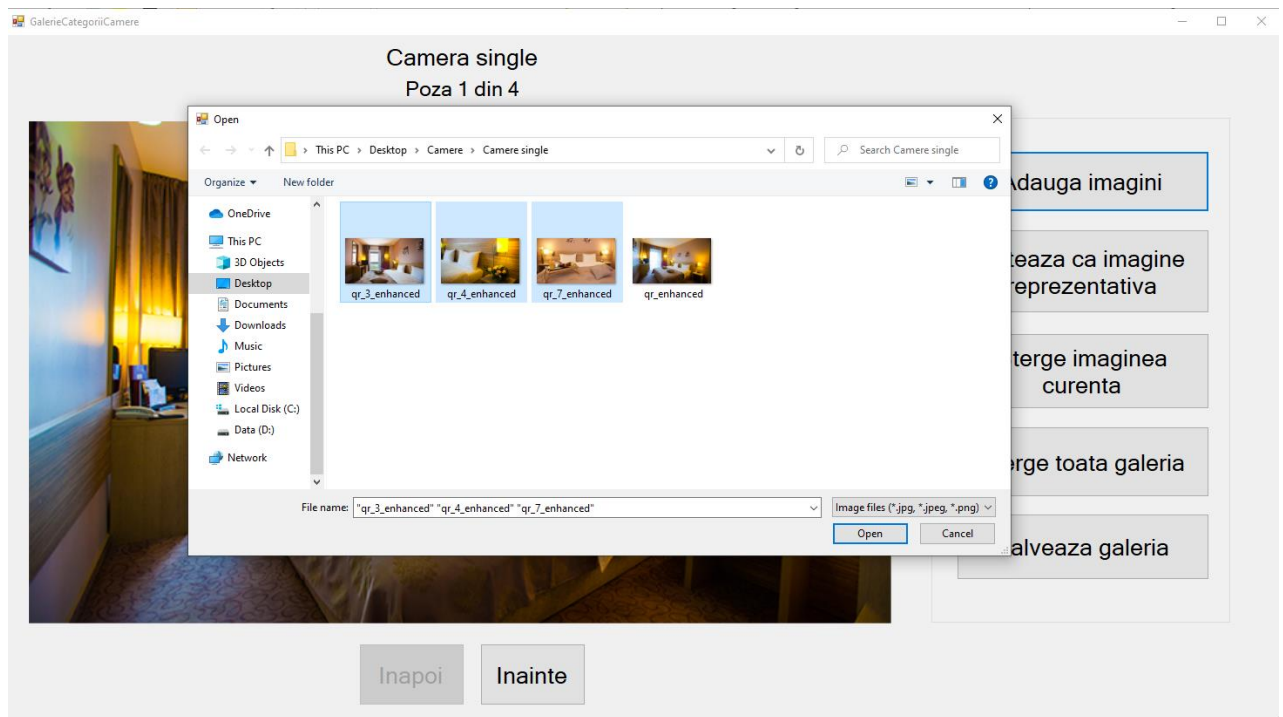


Figure 3.1. 11 Adding images to an image gallery

Set as representative image button will change the position of the current picture to position 1 (for example, if the current picture is "Picture 2 of 4", then it will become "Picture 1 of 4"). The first picture in a gallery is called the "representative picture" because in some forms in the application only this picture will be displayed. Thus, the administrator has the possibility to choose any image from the gallery as representative.

Delete current image button will remove the displayed image based on gallery navigation.

The Delete all gallery button it will delete all existing images in the gallery and leave a default image that all cameras have when recorded and have no images set.

Save gallery button will save all changes made (images added, images deleted or order changed).

These images are saved on the server in a file called *images* in which you can find both the images related to the room categories and the representative images of the services provided by the hotel. In the *image_category_rooms* table shown in figure 2.2.3.12, the path where the image is located is recorded. The name of the images is composed of the id related to the selected camera category and the image number, as can be seen in figure 3.1.12.

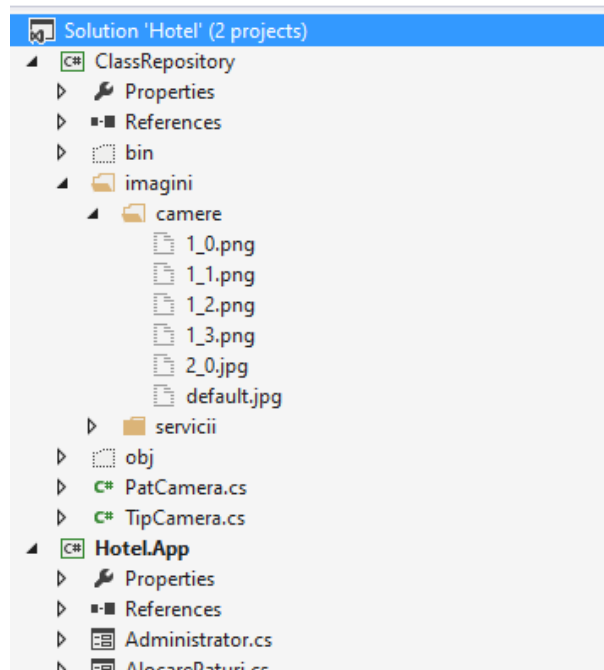


Figure 3.1. 12" images" file

The image *default.jpg* it is a preset that will not be deleted. All camera categories that do not have set images will have the default image.

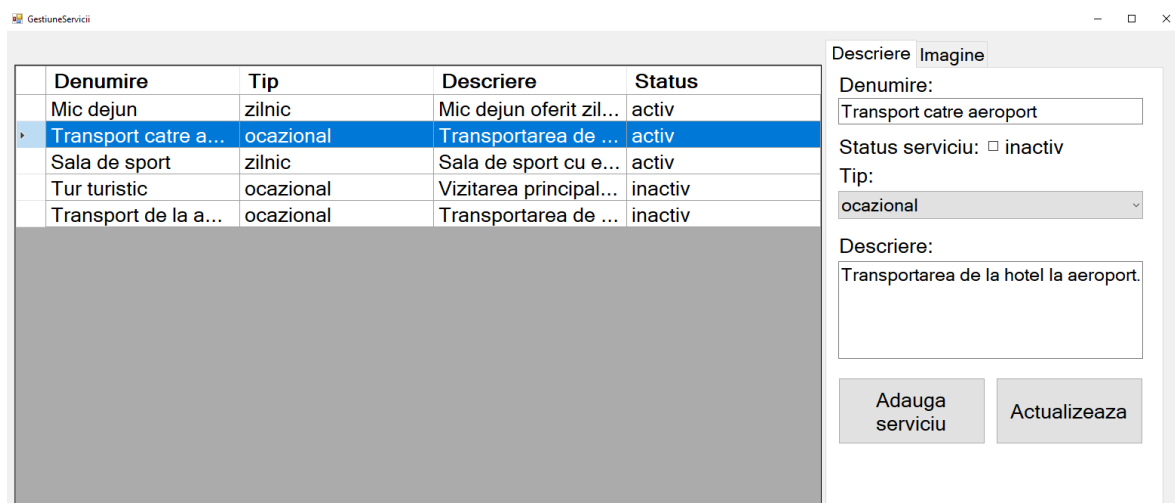


Figure 3.1. 13Service management window

Figure 3.1.13 shows the management window of the services provided by the hotel. In the table on the left side the current services are displayed, and on the right side there are the options to add a new service, to change the information of the selected service, as well as the possibility to change the representative image by accessing the *Image tab*.

When selecting a service, the data on the right side will be filled in automatically. The status indicates the availability of the service that allows its selection in the accommodation or reservation process if it is active. The type of service indicates how often it is made available. For example, occasional services are those services that the customer can benefit from on certain selected days, while daily services are made available every day.

Figure 3.1. 14The image setting form of the selected service

Figure 3.1.14 shows the buttons that allow updating the representative image of the selected service. This image will be displayed in other application windows. The image shown in the figure is the default when a new service is registered without setting an image. The algorithm works similar to that of setting camera category images, in that the images are saved in the *images* file with the name of the service id. The path to the file is saved in the database.

Change Image button will change the image displayed in the application, and the *Save* button will save the image to the server and update the file path in the database. The *Reset* button will reset the current image to the default one.

Tariff Management module will open a window containing two other buttons, named *Service Tariffs* and *Room Tariffs* . Here the admin can select the desired option which will open a new window.



Figure 3.1. 15Rate options window



Figure 3.1. 16Window for managing room rates

Figure 3.1.16 shows the hotel's room rates management module. On the left side you can find a list of all room categories registered in the hotel, and on the right side in the *Current Rate* tab you can find information such as: the representative image of the selected room category, the name of the category and the current rate (50 lei). On the right side of the tab there is a form for setting a new rate, which will become valid from the next day. After entering a new rate, the *Set new rate button* will register the new rate in the database, which will be valid starting from the current day until a new rate is set.

Tariffs tab shown in figure 3.1.17, you can see the history of the tariffs related to the category of rooms selected from the left list. This information is automatically updated when a room category is selected. The current rate is colored green to make it easier to highlight.

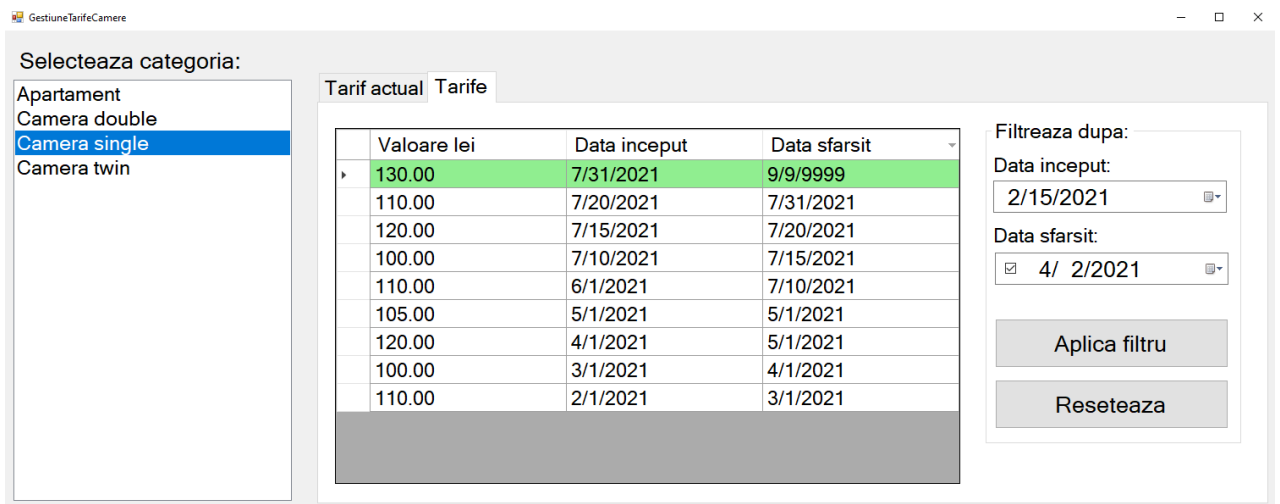


Figure 3.1. 17The " Tariffs" tab related to the tariff management module

At the same time, the tariff history can also be viewed in this table. On the right side there is a filter in which after entering the start and end dates, the desired period can be returned in which the administrator can view the history of rates, as shown in *figure 3.1.18*.

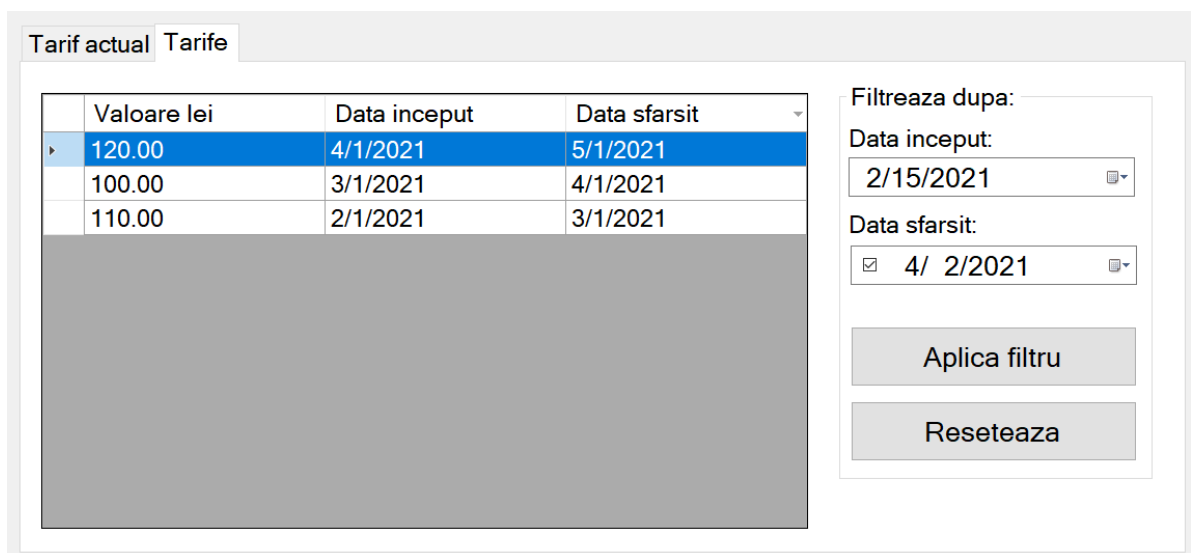


Figure 3.1. 18Filtering the room rate history

In the figure it can be seen that the selected period is February 15 - April 2. Thus, the algorithm returns the rates that are part of this range. This filtering is useful if there is a

history with several rates and the administrator wants to see the rates from a certain period of days.

The window for managing service rates can be seen in *figure 3.1.19* .

The screenshot shows a web application window titled 'GestiuneTarifeServicii'. On the left, under 'Selecteaza serviciul:', a list of services is shown with 'Mic dejun' selected. The main area has two tabs: 'Tarif actual' (active) and 'Tarife'. The 'Tarif actual' tab displays a representative image of a breakfast meal. Below the image, it shows 'Mic dejun' and 'Tarif: 30.00 lei'. On the right, the 'Seteaza tarif' section contains the text 'Tariful va deveni valabil incepand de maine.', a 'Tarif nou (lei):' input field with '0.00' entered, and a 'Seteaza tarif nou' button.

Figure 3.1. 19Window for the management of service tariffs

After selecting a service from the list on the left, the information on the right will be automatically filled in on the "Actual rate" tab (the representative image and the rate for the current day). As in the case of room rate management, on the right side there is a form that allows setting a new rate that will become valid starting the next day.

In the "Tariffs" tab, you can see the history of the service selected from the list, as illustrated in *the figure 3.1.20* .

The screenshot shows the same web application window, but with the 'Tarife' tab selected. The left sidebar remains the same. The main area displays a table of tariff history for 'Mic dejun'. The table has three columns: 'valoare_lei', 'data_inceput', and 'data_sfarsit'. It shows two entries: one with a value of 30.00 lei from 8/1/2021 to 9/9/9999, and another with a value of 2.00 lei from 7/17/2021 to 8/1/2021. On the right, the 'Filtreaza dupa:' section includes 'Data inceput:' and 'Data sfarsit:' filters, both set to 8/30/2021, with 'Aplica filtru' and 'Reseteaza' buttons below.

valoare_lei	data_inceput	data_sfarsit
30.00	8/1/2021	9/9/9999
2.00	7/17/2021	8/1/2021

Figure 3.1. 20Viewing the tariff history

On the right side there is a form where you can enter the period in which you want to view the rates. If the box for the end date is not checked, then all tariffs that start after the entered start date will be displayed when the *Apply filter button* is pressed .

The *Discount Management* button will open a window in which you will find two buttons, called *Customer Discounts* and *Seasonal Room Discounts* . The first button will open a form through which the administrator can manage the discounts related to the categories of rooms registered in the database. In *figure 3.1.21* you can see the appearance of the form.

The screenshot shows a web application window titled 'GestiuneReduceriCamera'. On the left, under 'Selecteaza categoria:', there is a list of room types: 'Apartament', 'Camera double', 'Camera single' (highlighted in blue), and 'Camera twin'. The main area has two tabs: 'Reducere actuala' (active) and 'Reduceri'. The 'Reducere actuala' tab displays a representative image of a single room, the text 'Camera single', a current discount of 'Reducere: 10%', and a validity period 'Valabila pana pe: 12/12/2021'. On the right, the 'Seteaza reducere' form contains fields for 'Discount nou (%)' (set to 0), 'Data inceput:' (8/30/2021), 'Data sfarsit:' (8/31/2021), and 'Durata:' (1 zile). A 'Seteaza reducere' button is at the bottom of this form.

Figure 3.1. 21Form for managing room discounts

On the left side is the list of camera categories registered in the database. In the "Current discount" tab on the right, you can find information about the camera (the representative image, the discount for the current day and the validity period of the discount). On the right side there is a form through which the administrator can set a new discount. He must enter the new discount percentage and the validity period. The duration in days shown below will automatically be filled. When you press the *Set discount button*, the new discount related to the selected room will be registered .

As in the rate management windows shown in *figures 3.1.17* and *3.1.20* , the "Reductions" tab contains a table where you can view the history and on the right side you can find the same form for filtering the results by the entered period.

Customer Discounts button in the previous menu will open a similar window where the customer category can be selected and the discount will be set for the desired period. There is also the possibility to view the history of discounts related to customer categories (students, pensioners, etc.).

In the administrators' menu, the *Manage personal data button* will open the form shown in *figure 3.1.22* through which it will be possible to update the personal data of the customers, in case they were entered incorrectly, as well as the login data of the hotel employees registered in the database data.

The screenshot shows a web application window titled "GestiuneDatePersonale". At the top, there is a label "Tip persoana:" followed by a dropdown menu currently showing "Client persoana fizica". Below this is a text input field labeled "CNP:" and a button labeled "Cauta dupa CNP". A section titled "Date personale" contains several input fields: "Nume:" and "Prenume:" (first and last names), "Email:", "Cod postal:" (postal code), "Telefon:" (phone number), and "Adresa:" (address). At the bottom of this section are two buttons: "Anuleaza modificarea" (cancel modification) and "Modifica date" (update data).

Figure 3.1. 22The form for the management of personal data

The ComboBox type control contains three values : " Individual customer", "Corporate customer" and "Employee". If the second option is selected, the first name field will disappear, the "CNP" field will be renamed to "CUI" and the button will have the text "Search by CUI".

In *figure 3.1.23* you can see the appearance of the form if the value " Employee" is selected.

The screenshot shows a web application window titled "GestiuneDatePersonale". The interface is divided into several sections. At the top left, there is a dropdown menu labeled "Tip persoana:" with "Angajat" selected. Below it is a text input field for "CNP:" and a button labeled "Cauta dupa CNP". To the right of this, there is a section titled "Cont de conectare angajat" containing two text input fields for "ID utilizator:" and "Parola noua:", and a button labeled "Modifica datele contului". Below these sections is a large box titled "Date personale". Inside this box, there are text input fields for "Nume:", "Prenume:", "Email:", "Cod postal:", "Telefon:", and "Adresa:". At the bottom of the "Date personale" box, there are two buttons: "Anuleaza modificarea" and "Modifica date".

Figure 3.1. 23" Employee" option selected in the personal data management form

On the right side, the "User ID" and "New password" fields were displayed because employees log in to the application using the credentials registered in the database and administrators have the possibility to modify the data.

When pressing the button *Search by CNP*, it will be checked if there is a person with the entered CNP and the available fields will be filled in and the *Cancel modification* and *Modify data* buttons will be activated , as can be seen in the figure 3.1.24 .

The screenshot shows a web application window titled "GestiuneDatePersonale". It contains two main sections: "Tip persoana:" and "Cont de conectare angajat".

Tip persoana: A dropdown menu is set to "Angajat". Below it is a text input field for "CNP:" containing "1991019176125". A button labeled "Cauta dupa CNP" is positioned below the CNP field.

Cont de conectare angajat: This section includes an "ID utilizator:" field with "andreip", a "Parola noua:" field, and a "Modifica datele contului" button.

Date personale: A larger section containing several input fields: "Nume:" (Popescu), "Prenume:" (Andrei), "Email:" (andrei@gmail.com), "Cod postal:" (600432), "Telefon:" (0723456789), and "Adresa:" (Bucuresti, str. Soarelui, nr. 8, bl. A2, ap. 10). At the bottom of this section are two buttons: "Anuleaza modificarea" and "Modifica date".

Figure 3.1. 24Personal data management form with completed information

When pressing the *Change data button* , it is checked if different information has been entered than the one returned, and the database will be updated. *The Cancel modification* button will clear all fields and re-enable search button by CNP. The *Change account data* button allows updating the account data of the selected employee.

The last button in the admin menu called *Reports* opens a form that generates graphs based on the option chosen. Figure 3.2.21 shows the window with the related controls.

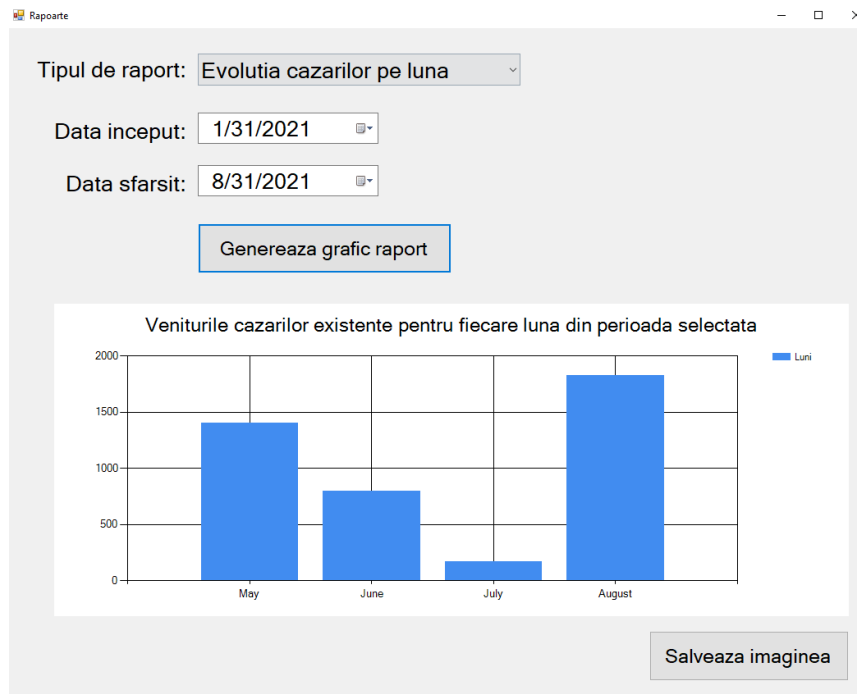


Figure 3.1. 25Form for generating graphs

The combobox control named *Report Type* contains several types of graphs that will generate different types of statistics. In the present case, the option to view the evolution of accommodation income per month was selected. In the *Start date* and *End date controls*, enter the period in which the administrator wants to see this statistic. When you press the *Generate graphic report button*, the image below will be automatically generated based on the data recorded in the database. The administrator has the possibility to save the image from the application in .png format. This will automatically save to the server file named *images*.

3.2 Receptionist Component

Through the Receptionist component, reservations and accommodations can be managed and their history can be viewed.

After logging into the application, if the account data belongs to a reception employee, the menu with related options will be displayed, as shown in the *figure 3.2.1*.

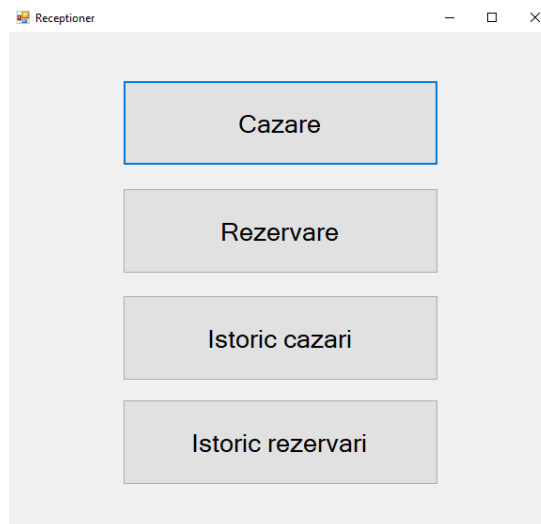


Figure 3.2. 1Receptionist menu window

Pressing the *Accommodation button* will open the accommodation form shown in *figure 3.2.2*.

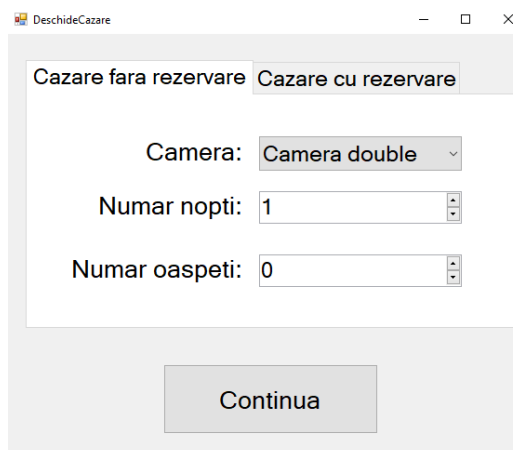
The image shows a software window titled "DeschideCazare". It has two tabs at the top: "Cazare fara rezervare" and "Cazare cu rezervare". The "Cazare cu rezervare" tab is selected. Below the tabs, there are three input fields: "Camera:" with a dropdown menu showing "Camera double", "Numar nopti:" with a text input field containing the number "1", and "Numar oaspeti:" with a text input field containing the number "0". At the bottom of the form is a button labeled "Continua". The window has standard Windows-style window controls in the top right corner.

Figure 3.2. 2Form for opening the accommodation window

Through this window, the necessary data will be entered to be able to check the availability of the rooms before they are displayed.

The first tab called *Accommodation without reservation* includes controls that will retrieve information such as: the desired room category, the number of nights the customer wishes to spend at the hotel and the number of guests he is accompanied with.

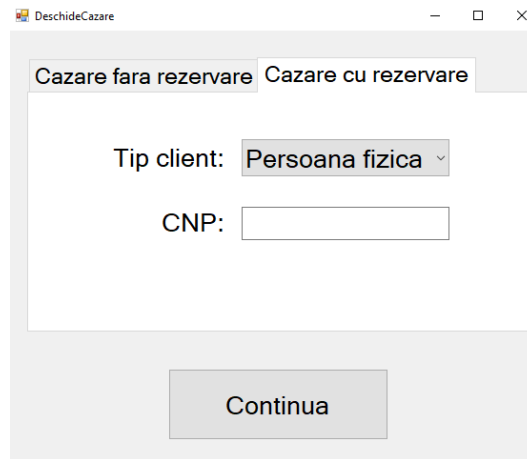
The image shows a software window titled 'DeschideCazare'. It has two tabs: 'Cazare fara rezervare' (selected) and 'Cazare cu rezervare'. Inside the selected tab, there is a form with the following elements: a label 'Tip client:' followed by a dropdown menu showing 'Persoana fizica'; a label 'CNP:' followed by an empty text input field; and a 'Continua' button at the bottom center.

Figure 3.2. 3Accommodation form with reservation

Figure 3.2.3 shows the form that will retrieve the client's CNP or CUI, depending on the selected type. Based on the identity code, it will be possible to check whether the customer has a reservation and the services and personal data of the customer, entered in the reservation, will be automatically completed.

When you press the *Continue button*, the room availability algorithm will be executed, which will take as parameters the start date of the stay and the end date. In the case of accommodation without reservations, the start date will be the current date.

The room availability algorithm determines the number of rooms available for the selected period. All accommodations and reservations that overlap with the client's stay period will be extracted from the database to determine if there are still free rooms for each room category and at the end only the categories selected by the client will be displayed. For the accommodations and reservations extracted, the number of rooms to be occupied is determined, and the algorithm will sort the accommodation period of the reservations in order to determine the minimum number of rooms required to accommodate all guests. This algorithm allows the hotel to accommodate as many customers as possible in as few rooms as possible, with the ability to select any available room desired by the customer. From the total number of hotel rooms, the number of rooms required to accommodate customers with reservations will be subtracted and the number of free rooms remaining will be determined.

After the execution of the availability algorithm, the message in *figure 3.2.4* will be displayed by which the receptionist is informed of the maximum number of rooms in the selected category that he can select and the accommodation window shown in *figure 3.2.5* will open.

Not all rooms displayed can be selected because a number of rooms are reserved for customers who are due to stay during the entered period, as calculated in the room availability algorithm.

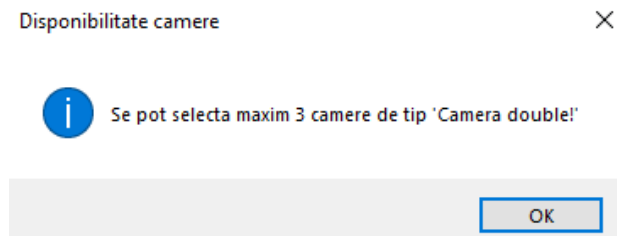


Figure 3.2. 4Information message about room availability

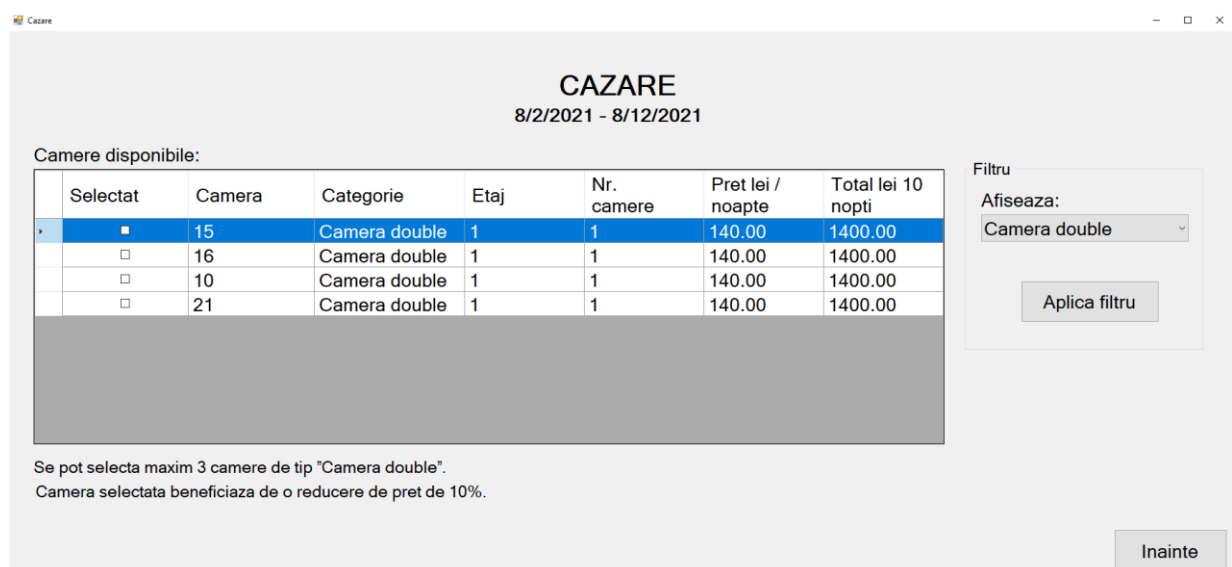


Figure 3.2. 5Accommodation window

On the left side is the list of available rooms for the selected period, from which the customer can choose. Information such as: the room number, the related category, the floor where it is located, the number of rooms, the price per night and the total cost for the number of nights can be found.

On the right side in the filter section there is a combobox control that will allow the receptionist to select another room type, in case there are customers such as individuals who want to occupy rooms for more guests.

After pressing the *Next* button, the page for selecting services will be displayed, shown in *figure 3.2.6*.

The screenshot shows a web application window titled "CAZARE" with the date range "8/2/2021 - 8/12/2021". The main section is "Servicii selectate:" which contains a table with columns: Serviciu, Data inceput, Data sfarsit, Pret lei / zi, and Total lei. The table is currently empty. To the right of the table is a "Sterge serviciul selectat" button. On the right side of the window, there is a section titled "Adauga serviciu:" containing a dropdown menu for "Serviciu:" (currently showing "Mic dejun"), date pickers for "Data inceput:" (8/ 2/2021) and "Data sfarsit:" (8/12/2021), and price information: "Pret zilnic: 15.00 lei" and "Pret total: 165.00 lei". Below this is an "Adauga serviciu" button. At the bottom right, there are "Inapoi" and "Inainte" buttons.

Figure 3.2. 6Accommodation window

at the bottom , such as the maximum number of rooms that can be selected and if there are any discounts for the selected room.

Each time a camera is selected, it is checked whether other cameras can be selected. When they can no longer be selected, the receptionist will receive the message " You cannot select more rooms", shown in *figure 3.2.7*.

The screenshot shows a window titled "Camere disponibile:". It contains a table with columns: Selectat, Camera, Categorie, Etaj, Nr. camere, Pret lei / noapte, and Total lei 10 nopti. The table has four rows. The first two rows are highlighted in orange and blue respectively, and both have their "Selectat" checkbox checked. The last two rows have their "Selectat" checkbox unchecked. A modal dialog box is open in the center of the table, displaying the message "Nu poti selecta mai multe camere" and an "OK" button. Below the table, there is a text area with the following information: "Se pot selecta maxim 1 camere de tip "Camera double".", "Camera selectata beneficiaza de o reducere de pret de 10%."

Figure 3.2. 7 Selection of rooms for accommodation

After pressing the *Next button* , the service selection page will be displayed, shown in figure 3.2.8 .

Figure 3.2. 7Accommodation services selection page

On the left side are the services entered by the receptionist. *The Start Date and End Date* columns indicate the period in which the customer wishes to benefit from these services. The rate is calculated according to the number of nights and the number of guests. In this case no number has been entered, but for accommodations where there is, the total price will be multiplied by the number of guests.

Add service button will add the service and the entered data to the list. *Delete selected service* button will delete the added service from the list if the customer changes his mind or the receptionist has entered the wrong data.

After pressing the *Next button* , the next page containing the customer registration form, shown in *figure 3.2.8*, will be displayed .

Informatii client:

Tip client:	<input type="text" value="Persoana fizica"/>	Nume:	<input type="text"/>	Prenume:	<input type="text"/>
CNP:	<input type="text"/>	Categorie:	<input type="text" value="Niciuna"/>	Cod postal:	<input type="text"/>
<input type="button" value="Cauta client dupa CNP"/>		Email:	<input type="text"/>		
		Telefon:	<input type="text"/>		
		Adresa:	<input type="text"/>		

Figure 3.2. 8Customer registration form

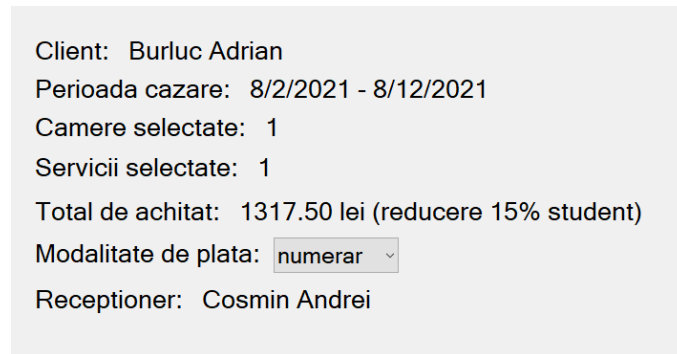
If the client has had reservations or accommodations before, it means that he already exists in the database and this can be checked after entering the identity code and pressing the *Search client by CNP button* . A message will be displayed if there is a registered customer with the entered ID code, and the fields will be filled in automatically, after which they will become inactive as shown in *figure 3.2.9 and the Delete data* button will appear if the completed data is not match or a wrong PNC or CUI was entered.

Informatii client:

Tip client:	<input type="text" value="Persoana fizica"/>	Nume:	<input type="text" value="Burluc"/>	Prenume:	<input type="text" value="Adrian"/>
CNP:	<input type="text" value="1991019176123"/>	Categorie:	<input type="text" value="student"/>	Cod postal:	<input type="text" value="1234567"/>
<input type="button" value="Cauta client dupa CNP"/>		Email:	<input type="text" value="adrian@gmail.com"/>		
<input type="button" value="Sterge datele"/>		Telefon:	<input type="text" value="0723456789"/>		
		Adresa:	<input type="text" value="Galati, Galati, str. Otelarilor, nr. 8"/>		

Figure 3.2. 9Completed form following the CNP verification

After pressing the *Next button*, the last page will be displayed, where you will find a summary of the selected rooms and services, as well as additional information such as the discounts applied for the customer category, if any, and the payment method, as illustrated in *figure 3.2 .10* .



Client: Burluc Adrian
Perioada cazare: 8/2/2021 - 8/12/2021
Camere selectate: 1
Servicii selectate: 1
Total de achitat: 1317.50 lei (reducere 15% student)
Modalitate de plata:
Receptioner: Cosmin Andrei

Figure 3.2. 10The information displayed on the last accommodation page

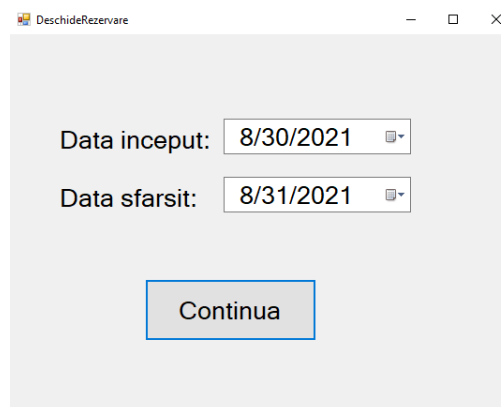
The discount for the customer category, in the present case for the student, is determined at the time of opening the accommodation window. For the selected period, the discounts for rooms as well as those for customer categories will be checked.

Discounts for customer categories apply to both rooms and selected services.

Upon pressing the *Confirm button*, the accommodation will be registered with the rooms and related services.

For accommodation with reservation, the process is similar, the difference being that the CNP will be entered in the form for opening the accommodation in *figure 3.2.3* and if there is a registered reservation, the accommodation window will open and the services and personal data of the client will be filled in automatically. The receptionist will only have to select the desired rooms and confirm the accommodation.

The *Reservation* button in the receptionists menu will open a form that contains two controls of the DateTime type in which the start and end date of the period in which the customer wants to stay at the hotel will be retrieved, as shown in the *figure 3.2.11*.



DeschideRezervare

Data inceput:

Data sfarsit:

Figure 3.2. 11Booking window opening form

After selecting the data, the room availability algorithm will be executed to determine the number of rooms that can be selected, after which the booking window will open with the first page where the rooms will be selected. In *figure 3.2.12*, 2 Single Camera type cameras have been added .

The screenshot shows a web interface titled "REZERVARE" with the date range "8/30/2021 - 8/31/2021". On the left, under "Camere selectate:", there is a table with the following data:

Camera	Nr. camere	Pret lei camera / noapte	Total lei 1 nopti
Camera double	2	140.00	280.00

Below the table is a large grey rectangular area. At the bottom right of the table area is a button labeled "Sterge camera selectata". On the right side of the interface, there is a sidebar titled "Adauga camera". It contains a "Tip camera:" dropdown menu currently set to "Camera double", a "Nr. camere:" numerical input field set to "3", and a "Tarif:" label with the value "140.00 lei". Below these is a "Pret total:" label with the value "420.00 lei". At the bottom of the sidebar is a button labeled "Adauga camera".

Figure 3.2. 12Room selection page for booking

On the right side you can select the desired room type and in the numerical counter *No. rooms*, enter the desired number of rooms. The selected room rate and the total value for the selected number of rooms are displayed below . Finally, after pressing the *Add room button* , the information will be added to the table on the left, where the total rate will be calculated according to the number of nights.

As in the accommodation form, there are *Back* and *Forward buttons* to help the receptionist navigate through the pages. The following pages are the same as in the case of accommodation, that is, the page for selecting services with the same controls and the same layout will be displayed, followed by the page for entering the customer's personal data, and finally the page for confirming the reservation is displayed, as shown in the *figure 3.2.13* .

REZERVARE

8/30/2021 - 8/31/2021

Client: Burluc Adrian
 Perioada rezervare: 8/30/2021 - 8/31/2021
 Camere selectate: 2
 Servicii selectate: 0
 Total de achitat: 238.00 lei (reducere 15% student)
 Receptioner: Cosmin Andrei

Inapoi Confirma

Figure 3.2. 13The final page of the booking window

Accommodation History button in the receptionists menu will open the form illustrated in *figure 3.2. 14* where the client's CNP will be entered to view the accommodation history.

DeschideIstoricCazari

Cauta istoric cazari

Tip client:

Persoana fizica

CNP:

1991019176123

Cauta cazari dupa CNP

Nume client	Checkin	Checkout	Mod. plata	Total achitat lei	Nume receptioner
Burluc Adrian	8/26/2021	8/27/2021	numerar	170.00	Cosmin Alex...
Burluc Adrian	8/27/2021	8/28/2021	numerar	84.15	Cosmin Alex...
Burluc Adrian	8/27/2021	8/29/2021	numerar	276.25	Cosmin Alex...

Vizualizeaza detalii cazare

Figure 3.2. 14Viewing the accommodation history

You will find information about the client's name, check-in and checkout date, payment method, the total amount paid and the name of the receptionist who confirmed the accommodation.

When you press the *View accommodation details button* , the accommodation form will open in which you will find all the information of the selected accommodation (details

about the selected rooms and services, the customer's personal data and the name of the receptionist who confirmed the accommodation).

In *figure 3.2.15* it can be seen how the accommodation window contains the information about the room selected between August 26 and August 27.

ISTORIC CAZARE
8/26/2021 - 8/27/2021

Camere selectate:

Selectat	Camera	Categorie	Etaj	Nr. camere	Pret lei / noapte	Total lei 1 nopti
<input checked="" type="checkbox"/>	11	Camera double	1	1	140.00	140.00

Camera selectata beneficiaza de o reducere de pret de 10%.

Inainte

Filtru
Afiseaza:
Camera double
Aplica filtru

Figure 3.2. 15Viewing the page for the selected rooms in the history of an accommodation

Pressing the *Next button* will take you to the services page, shown in *figure 3.2.16*.

ISTORIC CAZARE
8/26/2021 - 8/27/2021

Servicii selectate:

Serviciu	Data inceput	Data sfarsit	Pret lei / zi	Total lei
Mic dejun	8/26/2021	8/27/2021	30.00	30.00

Adauga servicii:
Serviciu:
Mic dejun
Data inceput:
8/26/2021
Data sfarsit:
8/27/2021
Pret zilnic: 30.00 lei
Pret total: 30.00 lei

InapoiInainte

Figure 3.2. 16Viewing the page for the selected services in the history of an accommodation

In *Figures 3.2.15 and 3.2.16* it can be seen that the controls for selecting rooms and services have been disabled. On the right side of the page for the selected services, the tariff valid for the respective period is displayed.

Next, the page with the customer's personal data will be displayed. In *figure 3.2.17* it can be seen that there are no more buttons for customer search or data deletion because these data are unmodifiable.

The screenshot displays a web form titled "ISTORIC CAZARE" with the date range "8/26/2021 - 8/27/2021". Below the title, it says "Informatii client:". The form contains several input fields and dropdown menus for customer data. At the bottom right, there are two buttons: "Inapoi" and "Inainte".

ISTORIC CAZARE		
8/26/2021 - 8/27/2021		
Informatii client:		
Tip client:	Persoana fizica	Nume: Burluc
		Prenume: Adrian
CNP:	1991019176123	Categorie: student
		Cod postal: 1234567
	Email: adrian@gmail.com	Adresa: Galati, Galati, str. Otelarilor, nr. 8
	Telefon: 0723456789	
		Inapoi Inainte

Figure 3.2. 17 Viewing the page for customer data in the history of an accommodation

On the last page, shown in *figure 3.2.18* , you can see the details of the accommodation.

The screenshot displays a web form titled "ISTORIC CAZARE" with the date range "8/26/2021 - 8/27/2021". Below the title, it lists various details of the accommodation, including the client's name, the period, the number of rooms and services selected, the total amount paid, the payment method, and the receptionist's name.

ISTORIC CAZARE	
8/26/2021 - 8/27/2021	
Client:	Burluc Adrian
Perioada cazare:	8/26/2021 - 8/27/2021
Camere selectate:	1
Servicii selectate:	1
Total de achitat:	170.00 lei
Modalitate de plata:	numerar
Receptioner:	Cosmin Andrei

Figure 3.2. 18 Viewing the accommodation details page in the history of an accommodation

There is also the possibility to view the reservation history by pressing the *Reservation history button* from the receptionists menu shown in *figure 3.2.1*.

will open , as can be seen in *the figure 3.2.19* .

The window titled 'DeschideIstoricRezervari' contains a search form on the left and a table of reservations on the right.

Search Form:

- Cauta istoric rezervari**
- Tip client:** Persoana fizica (dropdown menu)
- CNP:** 1991019176123 (text input)
- Cauta rezervari dupa CNP** (button)

Reservations Table:

Nume client	Data rezervarii	Checkin	Checkout	Nume receptioner
Burluc Adrian	8/27/2021 1:57 PM	8/28/2021	8/29/2021	Cosmin Alexan...

A large grey rectangular area is present below the table header, likely for displaying more reservation details. A button labeled 'Vizualizeaza detalii rezervare' is located at the bottom right of the window.

Figure 3.2. 19Window for viewing a customer's reservations

You can see information such as: the customer's name for the CNP entered, the date and time the reservation took place, the desired check-in and checkout dates and the name of the receptionist who confirmed the reservation.

When you press the *View reservation details button* , the reservation management form will open, with certain controls disabled because the object of the window is to view the data for that period .

In *figure 3.2.20* the first page of the booking history window is displayed.

The window titled 'ISTORIC REZERVARE' displays reservation details for the period 8/28/2021 - 8/29/2021.

Camere selectate:

Camera	Nr. camere	Pret lei camera / noapte	Total lei 1 nopti
Camera double	2	140.00	280.00

A large grey rectangular area is present below the table, likely for displaying more reservation details.

Summary Panel (Adauga camera):

- Tip camera:** Camera double (dropdown menu)
- Nr. camere:** 2 (input field)
- Tarif:** 140.00 lei
- Pret total:** 280.00 lei

An 'Inainte' button is located at the bottom right of the window.

Figure 3.2. 20Viewing the page for selected rooms in the history of a reservation

The last pages for viewing services and booking details work in the same way as the accommodation history pages. Controls are disabled to prevent data modification.

Chapter 4. Conclusions and proposals

The hotels of the current period manage to satisfy the needs of the customers in a faster and more efficient way thanks to the computer systems. The purpose of this work was oriented towards the design and implementation of an IT system to manage the reservation and accommodation processes of hotel customers. It offers a number of advantages in terms of streamlining the way in which the number of available rooms is determined and the possibility of accommodating as many customers as possible in as few rooms as possible, but at the same time giving customers the opportunity to choose the desired rooms at the time of accommodation.

The design and implementation of a website would allow customers to book rooms without the need for the presence of a receptionist, thus contributing to the process of automating the activities within the hotel. At the same time, through a website, more details could be presented regarding how the rooms are presented, information about beds, utilities, services, rates, etc. This information could be available to customers at any time without the need to contact an employee to provide more details.

Bibliography

1. Georgescu, C., *The relational and object approach in the analysis of computer systems* , RA Didactic and Pedagogical Publishing House, Bucharest, 2002;
2. Lupașcu, A., *Introduction to C#* – course notes, Galati, 2019;
3. Lupașcu, A., *Object- oriented programming* – course notes, Galati, 2019;
4. Negrescu, L. , *C# language for beginners - Volume VI - Visual Studio C# programming environment*, Albastra Bucharest Publishing House, 2011;
5. Niță, A., Niță, M., *Introduction to .Net Framework* – course notes, Bucharest, 2008;
6. Popa, G., *Microsoft SQL Server* , Economica Publishing House, 2006;
7. Solis, D., *Illustrated C#*, Apress Publishers, New York, 2010;
8. Skeet, J., *C# in Depth: Fourth Edition*, Manning Publications, 2019.

Web References

1. <https://www.researchgate.net/publication/235250226> A study of hotel information technology applications
2. <https://www.bngkolkata.com/hotel-computer/>
3. <https://www.nou.edu.ng/sites/default/files/2019-07/HCM432.pdf>
4. <https://techmonitor.ai/what-is/what-is-sql-server-4914415>
5. <https://docs.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-5.0>
6. <https://www.aspsnippets.com/Articles/Windows-Forms-WinForms-OpenFileDialog-Box-Tutorial-with-example-in-C-and-VBNet.aspx>

Annexes

Appendix 1 – Source code of the room availability algorithm for determining the minimum number of rooms required to be occupied during the period sent as a parameter.

```
public Dictionary<string, Int32> disponibilitateCamere(DateTime data_inceput, DateTime data_sfarsit)
{
    Dictionary<string, Int32> camereDisponibile = new Dictionary<string, Int32>();

    string data_inceputString = data_inceput.ToString("yyyy-MM-dd HH:mm:ss");
    string data_sfarsitString = data_sfarsit.ToString("yyyy-MM-dd");

    try{
        sqlCon.Open();

        string sqlcmd = "select 'Cazare' as tip, (select denumire from categorii_camere"+
            " where Id=camere.id_categorie_camere) as categorie_camere, cazari.checkin as data_inceput, " +
            "cazari.checkout as data_sfarsit, 1 as numar_camere from camere_cazari "+
            "join camere on camere_cazari.id_camera=camere.id_camera "+
            "join cazari on camere_cazari.id_cazare=cazari.id_cazare "+
            "where '" + data_inceputString + "' > checkin and '" + data_inceputString + "' < checkout "+
            "order by data_inceput asc";

        dataAdapter = new SqlDataAdapter(sqlcmd, sqlCon);
        dataAdapter.Fill(dataSet, "CamereOcupate");

        sqlcmd = "select 'Rezervare' as tip, (select denumire from categorii_camere "+
            "where Id = camere_rezervate.id_tip_camera) as categorie_camere, data_inceput, data_sfarsit, "+
            "camere_rezervate.numar_camere from rezervari "+
            "join camere_rezervate on rezervari.id_rezervare=camere_rezervate.id_rezervare "+
            "where '"+data_inceputString+"' <= data_inceput and '"+data_sfarsitString+"' > data_inceput "+
            "order by data_inceput asc";

        dataAdapter = new SqlDataAdapter(sqlcmd, sqlCon);
        dataAdapter.Fill(dataSet, "CamereRezervate");
    }
```

```

        sqlcmd = "select denumire from categorii_camere";
        dataAdapter = new SqlDataAdapter(sqlcmd, sqlCon);
        dataAdapter.Fill(dataSet, "CategoriiCamere");

        sqlcmd = "select denumire from camere join categorii_camere on categorii_camere.Id = camere.id_categorie_camere"+
            " where status='activ'";
        dataAdapter = new SqlDataAdapter(sqlcmd, sqlCon);
        dataAdapter.Fill(dataSet, "CategoriiCamereActive");

        dataSet.Tables["CamereOcupate"].Merge(dataSet.Tables["CamereRezervate"]);
    }
    catch(Exception err){
        MessageBox.Show(err.Message);
    }
    finally{
        sqlCon.Close();
    }
}

DataTable CamereOcupate = dataSet.Tables["CamereOcupate"].Clone();
foreach (DataRow cameraOcupata in dataSet.Tables["CamereOcupate"].Rows)
{
    if(Convert.ToInt32(cameraOcupata["numar_camere"]) > 1)
    {
        for (int i = 0; i < Convert.ToInt32(cameraOcupata["numar_camere"]); i++)
        {
            CamereOcupate.ImportRow(cameraOcupata);
        }
    }
    else
    {
        CamereOcupate.ImportRow(cameraOcupata);
    }
}

dataSet.Tables["CamereOcupate"].Columns.Add("alocat", typeof(bool));
CamereOcupate.Columns.Add("alocat", typeof(bool));

foreach (DataRow categorieCamere in dataSet.Tables["CategoriiCamere"].Rows)
{
    List<DataRow> camereOcupateDinCategorie = CamereOcupate.Select("categorie_camere='" +
        categorieCamere["denumire"].ToString() + "'").ToList();

    int nrCamereNecesare = 0;
    DateTime dataInceput;
    DateTime dataSfarsit;
    foreach (DataRow cameraOcupata in camereOcupateDinCategorie)
    {
        if (cameraOcupata["alocat"].ToString() != "True")
        {
            dataInceput = DateTime.Parse(cameraOcupata["data_inceput"].ToString()).Date;
            dataSfarsit = DateTime.Parse(cameraOcupata["data_sfarsit"].ToString()).Date;
            nrCamereNecesare++;

            cameraOcupata["alocat"] = true;

            foreach (DataRow cameraOcupata2 in camereOcupateDinCategorie)
            {
                if (cameraOcupata2["alocat"].ToString() != "True")
                {
                    if (DateTime.Parse(cameraOcupata2["data_inceput"].ToString()).Date >= dataSfarsit)
                    {
                        dataSfarsit = DateTime.Parse(cameraOcupata2["data_sfarsit"].ToString()).Date;

                        cameraOcupata2["alocat"] = true;
                    }
                }
            }
        }
    }
}

```

```

int nrCamereActive=0;
foreach (DataRow row in dataSet.Tables["CategoriiCamereActive"].Rows)
{
    if (row["denumire"].ToString() == categoriiCamere["denumire"].ToString())
    {
        nrCamereActive++;
    }
}

camereDisponibile.Add(categoriiCamere["denumire"].ToString(), nrCamereActive - nrCamereNecesare);
}

return camereDisponibile;
}

```

Appendix 2 – Algorithm for recording in the database the data entered in the accommodation form.

```

try
{
    con.Open();
    SqlCommand cmd;
    if (idClientGasit == null)
    {
        sqlcmd = "insert into clienti output inserted.id_client values ('"+tbEmail.Text+"', '"+
            tbTelefon.Text+"', '"+tbAdresa.Text+"', '"+tbCodPostal.Text+"', '"+cbTipClient.Text+"')";
        if (cbCategorieClient.Text == "Niciuna")
            sqlcmd = "insert into clienti output inserted.id_client values ('" + tbEmail.Text +
                "', '" + tbTelefon.Text + "', '" + tbAdresa.Text + "', '" + tbCodPostal.Text + "', NULL)";
        cmd = new SqlCommand(sqlcmd, con);

        idClientGasit = Convert.ToString(cmd.ExecuteScalar());

        if (cbTipClient.Text == "Persoana fizica")
            sqlcmd = "insert into persoane_fizice values ('"+idClientGasit+"', '"+tbNume.Text+"', '"+
                tbPrenume.Text+"', '"+tbCI.Text+"')";
        else if (cbTipClient.Text == "Persoana juridica")
            sqlcmd = "insert into persoane_juridice values ('" + idClientGasit + "', '" + tbNume.Text +
                "', '" + tbCI.Text + "')";
        cmd = new SqlCommand(sqlcmd, con);
        cmd.ExecuteNonQuery();
    }

    sqlcmd = "insert into cazari output inserted.id_cazare values ('" + idClientGasit + "', '" +
        dataInceputCazare + "', '" + dataSfarsitCazare + "', '" + cbModPlata.Text + "', '" +
        totalLei.ToString() + "', NULL)";
    cmd = new SqlCommand(sqlcmd, con);

    int idCazare = Convert.ToInt32(cmd.ExecuteScalar());
}

```

```

foreach (DataRow cameraSelectata in camereSelectate)
{
    sqlcmd = "insert into camere_cazari values('" + idCazare + "', '" +
        cameraSelectata["id_camera"].ToString() + "')";
    cmd = new SqlCommand(sqlcmd, con);
    cmd.ExecuteNonQuery();
}

foreach (DataRow serviciuSelectat in serviciiSelectate.Rows)
{
    sqlcmd = "insert into servicii_cazari values('" + idCazare + "', '" +
        serviciuSelectat["id_serviciu"].ToString() + "', '" + serviciuSelectat["data_inceput"].ToString() +
        "', '" + serviciuSelectat["data_sfarsit"].ToString() + "')";
    cmd = new SqlCommand(sqlcmd, con);
    cmd.ExecuteNonQuery();
}

MessageBox.Show("Inserat!");
}
catch (Exception err)
{
    MessageBox.Show(err.Message);
}
finally
{
    con.Close();
}
}

```

Appendix 3 – The source code related to the algorithm for adding an image to the image gallery of the camera category.

```

try
{
    con.Open();
    sqlcmd = "delete from imagini_categorii_camere where id_categorie=" + idCategorie;
    SqlCommand cmd = new SqlCommand(sqlcmd, con);
    cmd.ExecuteNonQuery();
    MessageBox.Show("delete db");

    sqlcmd = "insert into imagini_categorii_camere values(" + idCategorie + ", @Pozitie)";
    cmd = new SqlCommand(sqlcmd, con);

    cmd.Parameters.Add("@Pozitie", SqlDbType.NVarChar);

    string denumireImagine = @"_" + idCategorie + "_";
    string[] imaginiDeSters = System.IO.Directory.GetFiles(path + "\\ClassRepository\\imagini\\camere\\", denumireImagine);
    foreach (string pathImagine in imaginiDeSters)
    {
        DataRow imagineExistenta = galerieNoua.Select("imagine='" + pathImagine + "'").FirstOrDefault();
        if (imagineExistenta == null)
            System.IO.File.Delete(pathImagine);
        else
        {
            string newFileName = Path.GetFileNameWithoutExtension(pathImagine) + "temp" + Path.GetExtension(pathImagine);

            System.IO.File.Move(pathImagine, path + "\\ClassRepository\\imagini\\camere\\" + newFileName);
            imagineExistenta["imagine"] = path + "\\ClassRepository\\imagini\\camere\\" + newFileName;
        }
    }

    int pozitie = -1;
    foreach (DataRow imagineNoua in galerieNoua.Rows)
    {
        string extensie = imagineNoua["imagine"].ToString().Substring(imagineNoua["imagine"].ToString().LastIndexOf('.'));
        pozitie = pozitie + 1;
        System.IO.File.Copy(imagineNoua["imagine"].ToString(), path + "\\ClassRepository\\imagini\\camere\\" +
            idCategorie + "_" + pozitie + extensie);
    }
}

```

```

        cmd.Parameters["@Pozitie"].Value = "\\ClassRepository\\imagini\\camere\\" + idCategorie + "_" + pozitie + extensie;
        imagineNoua["imagine"] = path + "\\ClassRepository\\imagini\\camere\\" + idCategorie + "_" + pozitie + extensie;
        cmd.ExecuteNonQuery();
    }

    denumireImagine = @"*temp*";
    imaginiDeSters = System.IO.Directory.GetFiles(path + "\\ClassRepository\\imagini\\camere\\", denumireImagine);
    foreach (string pathImagine in imaginiDeSters)
    {
        System.IO.File.Delete(pathImagine);
    }

    galerie = galerieNoua.Copy();
    galerieModificata = true;
}
catch (Exception err)
{
    MessageBox.Show(err.Message);
}
finally
{
    con.Close();
}

```