Technical Report
NAO Challenge24

By

Team NaoNexus

For more information:

Visit our website, our repository
and write an email to: socialnaonexus@gmail.com

"Alle Stimate" High School, Verona

Abstract

This year's Nao Challenge is based on retail: the direct sale of goods or services to end consumers through physical points of sale or online.

The NaoNexus team therefore chose to use the NAO as a robotic assistant in the Swarovski store in order to make the choice of product to purchase easier and faster for everyone. Furthermore, the NAO is used to manage the quantities of products in stock per store and notify the Store Manager in case of need for replenishment.

In this report the working methods, the technologies used and the sub-teams of the NaoNexus team will be explained.

Index

1 Methodology and techniques used

Since NaoNexus is a large group, there was a need to keep track of *tasks* and keep each component up to date with the latest version of the developed software. For this purpose, the team leveraged two important technologies: *GitHub* and *Notion*.

1.1 GitHub

GitHub is a tool that helped the team save and manage code. It is a cloud-based Git repository hosting platform that helps members track changes made in the code thanks to version control. All changes made are tracked and can be rolled back if they lead to errors.

Through branching and merging, each member was able to safely work on the code, avoiding directly affecting the main version.

1.2 DevOps with Notion e Agile methodology

To work on the project and track progress, the team used Notion, a collaborative notes management platform. The term DevOps comes from the union of "development" and "operations" and consists of the various methods that aim to improve end-to-end collaboration. The Notion web-app features many tools such as task management, project progress tracking and to-do lists, thanks to

which each team member is aware of the tasks to be completed at each appointment. This collaboration platform integrates wikis and databases, creating a workspace for managing data and tasks.

The project was managed with the Agile methodology: the main tasks were divided into sub-tasks to be then distributed among the team members, leading to greater efficiency in software development.

2 The two roles of the NAO

The first NAO is the one that is among customers and is used as an assistant who can be contacted directly in the store. It is connected to a phone via an HTTP server, and users can start the conversation with the robot via an application they have previously connected to. After analyzing the customer using Morphcast technology, the NAO asks him for some details (the budget, if it is a gift and the gender of the recipient) and proposes a product. This is chosen via a decision tree that uses all the information it has on the recipient of the jewel. The dialogue with the customer continues until he chooses to add the product and its hypothetical combinations (always proposed by the NAO) to the cart.

The role of the second NAO, present at the cash register, is that of store management assistant. The web app is connected via an HTTP server to a database and allows Swarovski employees to view sales of a specific product, registered users, active carts, products on the shelf and in stock. The NAO has the role of alerting the Store Manager of the need to order products below a certain threshold in the warehouse (value obtained from the web app), including those in the window, which will be restocked as soon as possible.

2.1 NAO as shop assistant

The first NAO, already present in the store, begins the dialogue with the customer and analyzes it: calculates the level of anger, disgust, fear, happiness, neutrality, sadness, surprise and attention, detects whether it is a man or a woman and calculate a probable age. At this point the customer is asked if it is a gift for another person.

If the answer is yes, the humanoid requests additional information that it cannot detect on its own: the age and gender of the recipient. However, if it is a personal purchase, these steps are skipped as Morphcast technology is used.

In both cases, the budget and the type of jewel the customer is looking for (category) are requested. Using this data, through a decision tree, the NAO offers products to the user until he adds one to the cart. At that point, the customer is directed to the checkout with the QR code generated on the payment app.

For the operation of the first NAO robot, we needed:

- **Choregraphe:** In our project, we took an innovative approach to programming the NAO, avoiding the use of Choregraphe software, traditionally used to configure behaviors and interactions in humanoid robots.

  We chose instead to write the code in Python, which

interacts directly with the NAO. Initially, we extrapolated code blocks from Choregraphe, which, together with key components such as dialogue, decision tree and Morphcast technology, formed the basis for establishing interactive communication between the robot and the customer. Next, the blocks were integrated within custom Python functions, giving us the flexibility to program the various interactions of the NAO.

We implemented a feature to extract customer responses, allowing the robot to understand and react to human interactions. Other functions used are those of speech recognition and synthesis, through which the robot is able to convert the text into vocal responses, thus enriching its interaction with the user. Furthermore, we have implemented a function that allows the robot to detect the presence of a customer and follow him with his gaze. This not only facilitates user profiling, but adds a level of interactivity to the NAO, creating a more engaging experience for the end customer.

The Python code-based approach allowed us to customize the robot's interactions in detail and dynamically adapt them to the needs of the project. Our methodology aims to fully exploit the potential of the NAO robot, offering a unique and interactive experience to users.

The animated say function allows the NAO to vocally reproduce Python strings, while the face tracker function allows the NAO to follow the face of the human with whom it is conversing or interacting.

Here's how we implemented these Choregraphe blocks in Python:

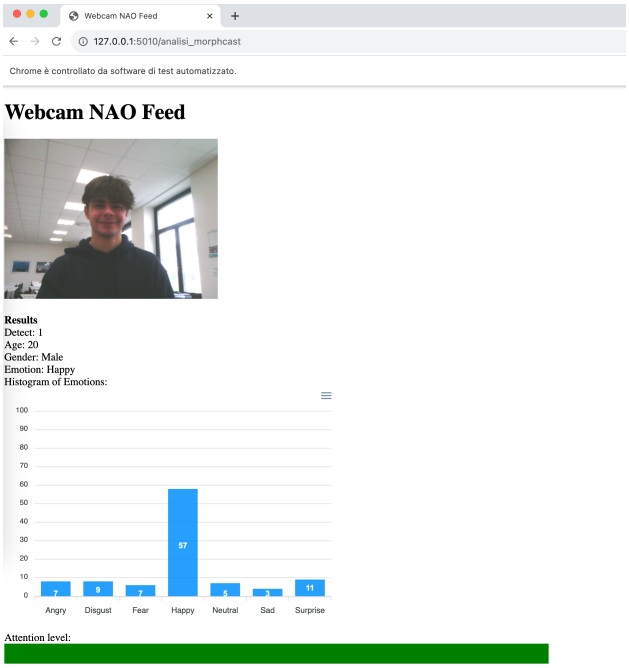
```
def nao_animatedSayText(text_to_say):
tts_proxy = ALProxy("ALAnimatedSpeech",\
nao_ip, nao_port)
animated_speech_config = {"bodyLanguageMode":\
"contextual"}
tts_proxy.say(text_to_say,\
animated_speech_config)
tts_proxy = None

def nao_face_tracker():
tracker_proxy = ALProxy("ALTracker", nao_ip,\
nao_port)
targetName = "Face"
faceWidth  = 0.1
tracker_proxy.setMode("Head")
tracker_proxy.registerTarget(targetName,\
faceWidth)
tracker_proxy.track(targetName)
```

- **HTTP Server:** The server is a very complex piece of code written in Python where all the project data is managed. It is connected to the database to extract and add data, allows the operation of the web app and the NAO and the analysis of images passed by the robot for the extraction of emotions.
- **NAO:** NAO is the central part of the project. It is connected to the HTTP server and analyzes the user through its camera, extracting and saving emotions. Interacts with the customer by recommending the most suitable item to choose. Furthermore, it notifies the Store Manager when the products on the shelf or in the warehouse are about to run out, specifying the model of the jewel.
- **Morphcast API:** Morphcast allowed us to use its AI API. For more information visit their website:

  MorphCast®

  Here is the web page of the web app which allows you to graphically view the work of the Morphcast technology:



- **Decision Tree:** The decision tree is a machine learning model that organizes logical decisions into a tree structure for classification or regression. Each node represents a choice based on an attribute, with branches indicating the possible consequences of that decision. Our decision tree takes into consideration three main variables that are given as input by the user:
  - Gender
  - Age
  - Budget

  Based on these variables, the code returns a list of

jewels that may be of interest to the user. Let's take a closer look at our code:

- With the first condition we check the gender of the person for whom the jewel is intended and based on the user's response we proceed to a specific branch of the decision tree (male/female).
- Requesting the age of the future owner of the jewel is necessary in order to recommend a product that may be more appreciated.
- When we enter the if and provide a budget, the program checks the price of each product through a succession of conditions, checking whether it meets the user's needs.

If the decision tree finds one or more products that match gender, age, jewelry category and budget, it outputs a list with the recommended ones.

Part of the decision tree code, written in Python:

```
.....................................................
if gender == 'male':
    if age > 60:
        return []
    elif age < 20:
        if category == 'bracelet':
            if budget >= 155:
```

```
                for item in product_info:
                    if item['gender'] == 'M'\
                    and item['age'] < 20 and \
                    item['category'] == 'bracelet'\
                    and item['prezzo'] >= 155:
                   gioielli_consigliati.append\
                   (item['id'])
                return gioielli_consigliati
            else:
                return []
        elif category == 'necklace':
            if budget < 175:
                return []
            else:
                for item in product_info:
                    if item['gender'] == 'M' and \
                    item['age'] < 20 and \
                    item['category'] == 'necklace'\
                    and item['prezzo'] >= 175:
                    gioielli_consigliati.append\
                    (item['id'])
                sreturn gioielli_consigliati
        else:
            return []
.....................................................
```

- **Database:** To manage product, customer, and order information, we created a database using PostgreSQL via the pgAdmin GUI.

ER Model (Entity-Relationship)

Before proceeding with the implementation of the database, we performed a preliminary ER model design phase to identify key entities and relationships within the system.

This model helps to visualize the structure of the database and the relationships between the entities involved.

The main entities identified in the ER model include:

  - Customer: represents the store's customers, with attributes such as ID, username, name, surname, password
  - Cart: Represents information regarding customer carts, with attributes such as ID and Customer ID
  - Subject: Represents the various jewelry items offered by the store, with attributes such as ID, category, description, price, photo, etc.
  - Cart Object: Represents the objects in the cart, with attributes such as ID, Cart ID and Item ID

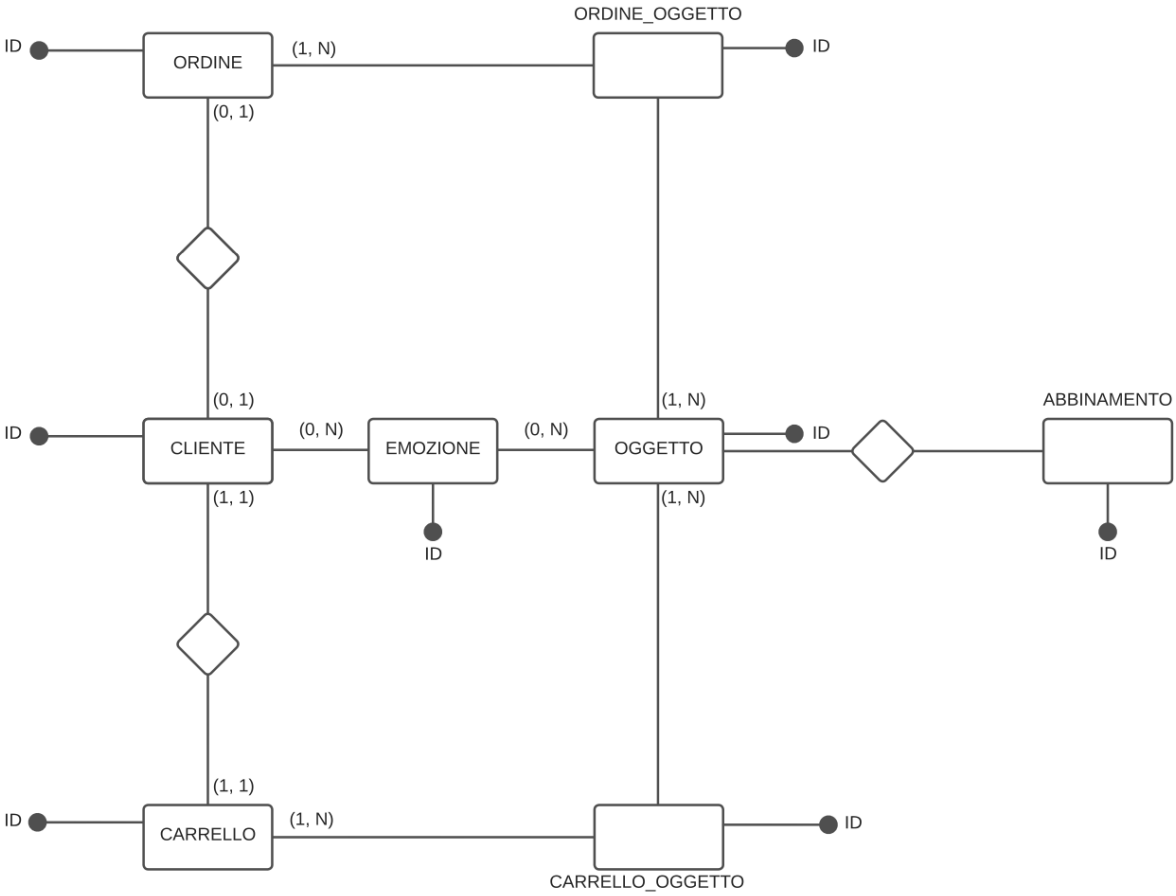
- Order: Represents orders placed by customers, with attributes such as ID, customer ID, purchase date, payment method, etc.
- Order Subject: Represents the specific details of each order,
  including: order ID, item ID, quantity, unit price, etc.
- Emotion: represents the customer profile and his emotions, with attributes such as ID, age, gender and satisfaction rating
- Match: Represents the possible matches for each product, with attributes such as first item ID, second item ID, etc.

Relational Logical Diagram

Based on the designed ER model, we created a logical and relational schema to implement the database using tables in PostgreSQL.

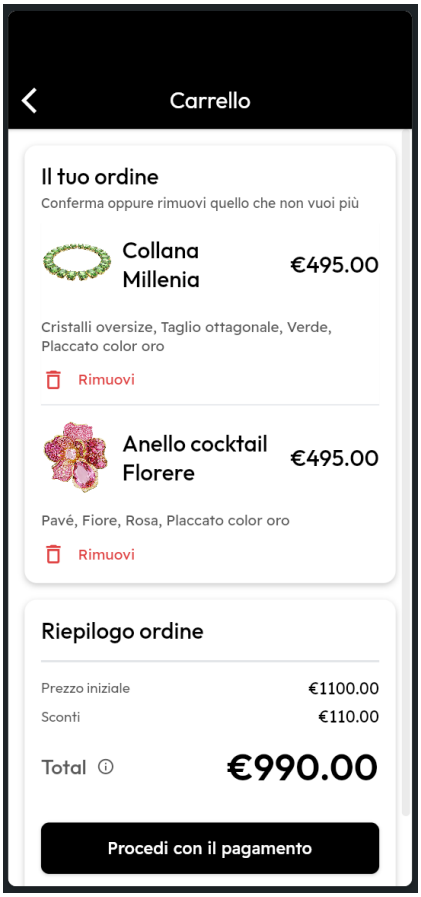
Relationships between tables were implemented using primary keys and foreign keys to ensure referential integrity and correct data association.

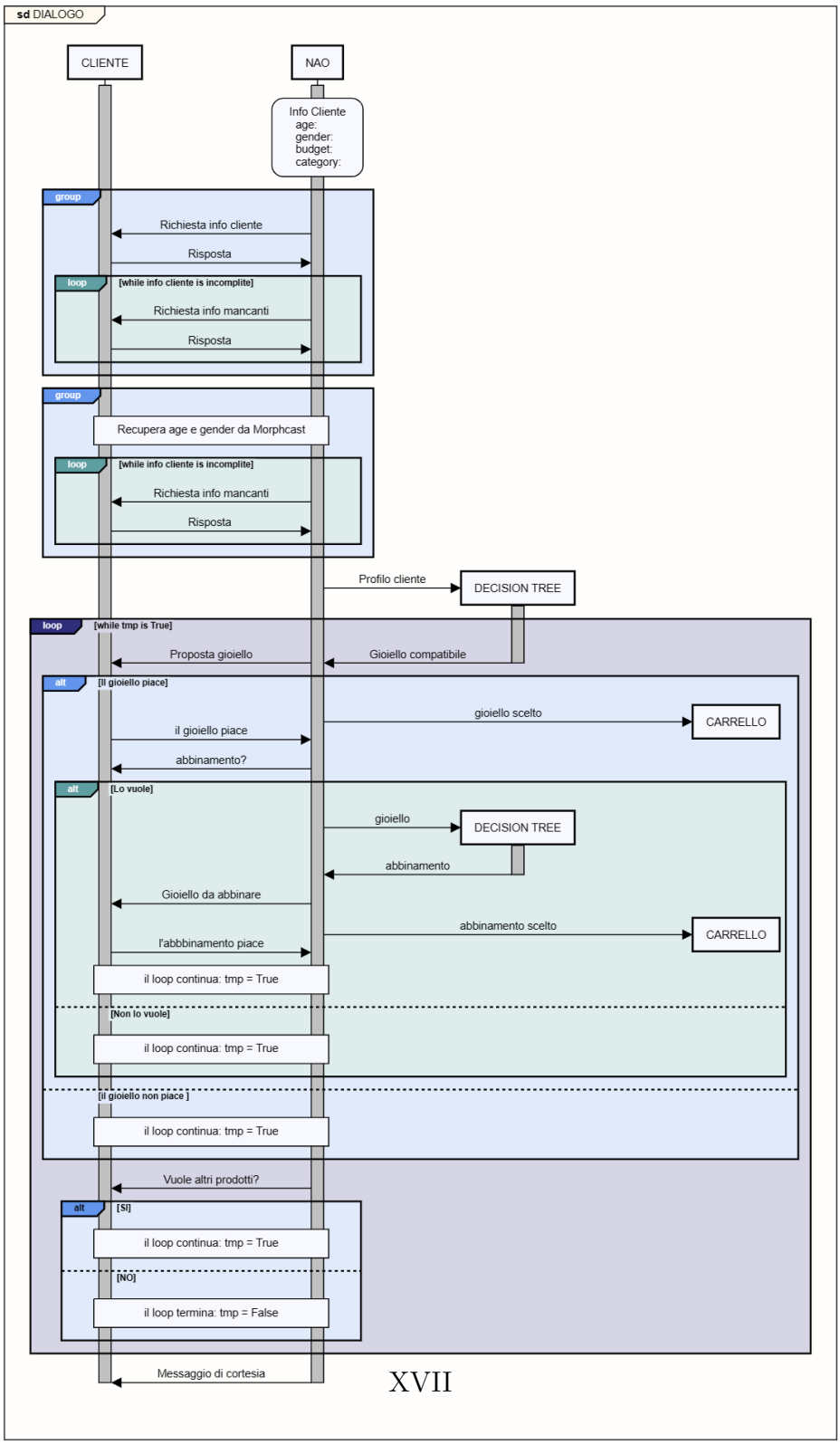
To ensure that the database worked as expected, during its creation we populated the tables with sample data and tested the queries.



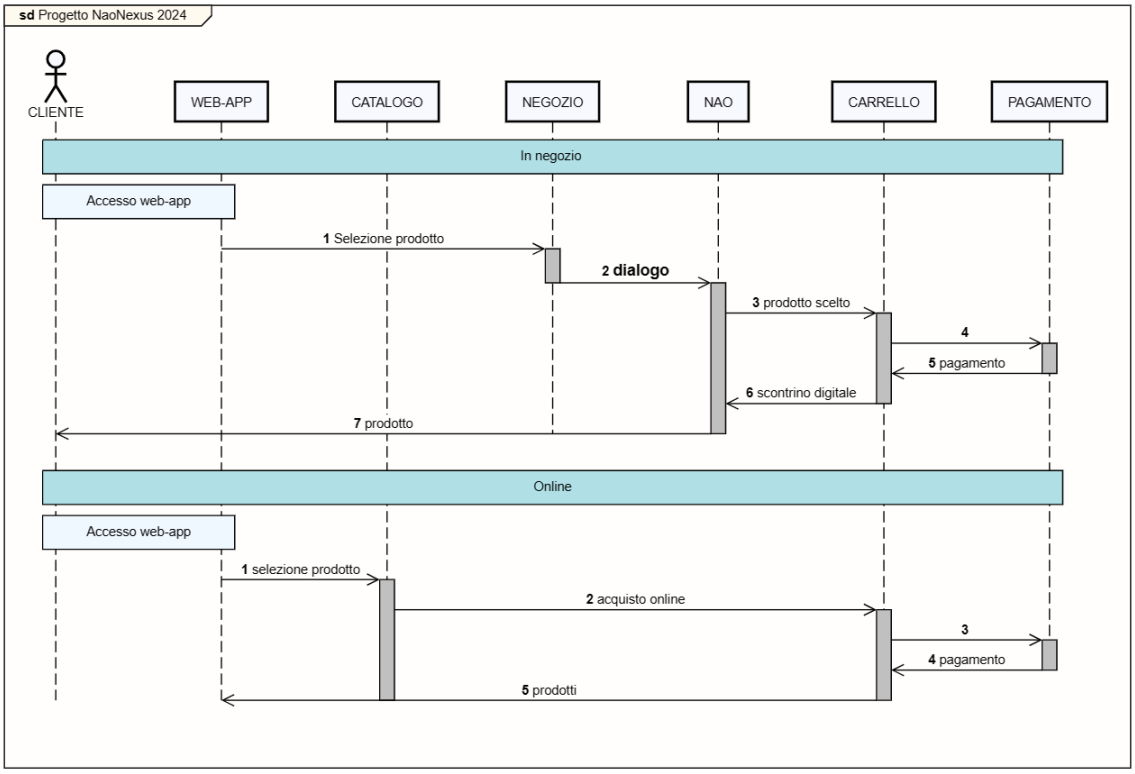
- **App:** The app is the tool with which the end user can interface with OneRetail. It gives access to various functionalities, such as a catalogue for online shopping, as well as the interactive store activities. it was developed in Flutter, a framework that compiles binaries for both Android and iOS, cutting down development time and increasing compatibility. The app communicates directly with the web server through Rest API calls, sharing data in the JSON format.

This way, it can receive data such as the catalogue or the cart linked to the logged user, as well as send requests like adding and removing an item to the cart, or even activating the NAO itself.




sd DIALOGO
CLIENTE
NAO
Info Cliente
age:
gender:
budget:
category:
group
Richiesta info cliente
Risposta
loop
[while info cliente is incomplite]
Richiesta info mancanti
Risposta
group
Recupera age e gender da Morphcast
loop
[while info cliente is incomplite]
Richiesta info mancanti
Risposta
Profilo cliente
DECISION TREE
loop
[while tmp is True]
Proposta gioiello
Gioiello compatibile
alt
[Il gioiello piace]
gioiello scelto
CARRELLO
il gioiello piace
abbinamento?
alt
[Lo vuole]
gioiello
DECISION TREE
abbinamento
Gioiello da abbinare
abbinamento scelto
CARRELLO
l'abbbinamento piace
il loop continua: tmp = True
[Non lo vuole]
il loop continua: tmp = True
[il gioiello non piace]
il loop continua: tmp = True
Vuole altri prodotti?
alt
[SI]
il loop continua: tmp = True
[NO]
il loop termina: tmp = False
Messaggio di cortesia

Description: The NaoNexus 2024 project orchestrates a seamless shopping experience, combining physical and online interactions. Customers, represented by CUSTOMER, can use a web application (WEB-APP) to access a catalog (CATALOGUE) and make product selections. In the physical store, SHOP enters into a dialogue with the NAO robot, which adds the chosen products to the shopping cart (CART). Payment is processed via CHECKOUT and a digital receipt is generated. Similarly, in the online scenario, customers access the WEB-APP, interact with the CATALOG and select items added to the online CART. The payment process is managed by PAYMENT and customers receive purchased products and information online. This integrated approach improves the purchasing path for NaoNexus in 2024.



The Dialogue Diagram illustrates an online purchasing dialogue between the customer (CUSTOMER) and NAO. Two purchase scenarios are considered: purchase as a gift and personal purchase. Using a decision tree, the assistant suggests compatible jewelry based on the customer's profile. The dialogue involves cycles to gather comprehensive information and make product selections. The interaction continues until the customer completes the purchase or decides not to add any more items. Next, a courtesy message is sent and both the customer and assistant are deactivated, concluding the dialogue.

2.2 NAO as management assistant

The role of the second NAO is to manage the quantity of jewelery on the shelf and in the warehouse, alerting the Store Manager when necessary so that he can replenish the missing jewellery. On the web app you can also see the sales of the various products individually, the active accounts and the active carts.

For the operation of the second NAO robot, we needed:

1. **NAO:** The NAO humanoid is essential to make shop management more humane and faster. Thanks to it there is no longer any need to check how many objects are in the warehouse or on the shelf as all you need to do is press a button on the web app.

   The NAO, once the "Report" button on the web app is pressed, will immediately notify the Store Manager to refill the shelf or place an order for the warehouse in case the products are below a certain threshold n.

   This is essential to make the management of jewelery quantities faster, thus saving time for the Store Manager.

2. **HTTP Server:** It is the central point where all the project data is managed, and it is a very complex piece of code written in Python. It allows the webapp

to function, is connected to the database to extract and add data, allows the NAO to function and allows the analysis of images passed by the NAO for the extraction of emotions.

3. **Web app:** An important part of the server is the webapp that is served by it: HTML pages are populated using Jinja and are sent using Flask. This library is responsible for the operation of the Rest server which also distributes data in JSON format. In this webapp you can view active users and carts, also allowing data to be modified; furthermore, in the form of a graph you can see the sales of the various products. The web app also allows you to insert new users of the application, who can therefore only be registered in the store.
   Below is one of the pages of the web app, it allows you to check the quantity of products in the store:

SWAROVSKI

Gioielli sullo scaffale:

40

Report

ID	Titolo	Categoria	Qtà Scaffale	Foto	
1	angelic bracelet	bracelet	1		
2	angelic necklace	necklace	1		
3	constella cocktail ring	ring	0		

This page of the web app allows the Store Manager to view a table showing the quantities of the various jewels in the store. The "REPORT" button allows you to ask the NAO robot which products need to be replenished on the shelf/window. The humanoid will immediately notify the Store Manager which products are needed from the warehouse.

3 Team members

The team was divided into two groups, the social team and the coding team.

3.1 TEAM CODING

The coding team worked intensely to develop the code necessary for the operation of the projects related to the NAO robot. The members of this team held various roles:

- Edoardo Polfranceschi - Team Leader: He coordinated the team's activities and took care of the creation of the server and the web app.
- Aurora Savoia: She contributed to the creation of the application with the product catalog, product pages and QR code scanner.
- Shenal Fernando: He developed the NAO control using Python and no longer Choregraphe.
- Antonio Galati: He tried his hand at developing the application.
- Giacomo Santi: He created the decision tree for automatic product selection.
- Chiara De Marchi: She created the database in SQL language.

3.2 TEAM SOCIAL

The social team was responsible for promoting the project through social channels and producing multimedia content. The main activities carried out by this team include:

- Alberto Rubini - Team Leader: He coordinated the activities of the social team and took care of the creation of the contents.
- Davide Masini: He helped create the NaoNexus team's 2024 website.
- Arianna Antonelli: She helped create the site and digital content.
- Laura Mascalzoni: She took care of the team's social networks and wrote texts for the project.

4 Conclusions and thanks

The Nao Challenge was an intense and educational experience for the entire team. We managed to develop two innovative projects that exploit the capabilities of the NAO robot in different areas, demonstrating our technical and organizational skills.

We warmly thank the Swarovski Italia team for supporting us by giving us the opportunity to collaborate with them, and equally we thank Morphcast for allowing us to use their technology to humanize the NAO-customer interaction even more.

We all members of the team particularly thank Professor Giovanni Bellorio (Il Supremo) for giving us the opportunity to participate in this engaging project. And we also thank him for following us in the realization of the project by teaching us about the world of team work.