

**FACULTY OF COMPUTER SCIENCE**

**Bachelor in Computer Science and Engineering**

# **Degree Thesis**

**“Purchase Information System Implementation for  
the Cockpit portal”**

**SONIA FREI**

## **Supervisor:**

Firstname: Johann

Lastname: Gamper

Signature:

## **Student**

Firstname: Sonia

Lastname: Frei

Signature:

Academic Year: 2013/2014

## Contents

Abstract.....	3
1. Introduction .....	4
1.1 The University and its Need for a Purchase Information System .....	4
1.2 Cockpit .....	5
1.3 Contributions .....	6
1.4 Organization of the Thesis .....	7
2 Current State of PIS.....	8
2.1 Programming Language .....	9
2.2 Interface.....	9
2.3 Problems .....	10
3 Porting the PIS to the Cockpit .....	11
3.1 Logic of the Program.....	12
3.2 Programming Language .....	14
3.3 Implementation Choices .....	14
3.3.1 System Architecture.....	15
3.3.2 The Code .....	15
3.3.3 The Layout.....	16
4 Discussion.....	19
4.1 Lessons Learned .....	19
4.2 Advantages of the New System .....	20
4.3 State of the New PIS .....	21
5 Conclusion.....	24
6 Acknowledgments.....	25
7 Bibliography .....	26
8 Table of figures.....	27

## Abstract

Over the last two decades the pace of technological change has increased so quickly that software developed 20 years ago seem now primitive and coarse. With the foundation of the Free University of Bolzano 17 years ago, some applications essential for its operations have been developed. During time, many of them were completely redesigned to fit new requirements and trends: this never happened, however, for the Purchase Information System.

The main function of PIS is to manage employees' requests for new purchases of goods and services. Moreover, it is also used to delegate a co-worker during a user's absence time and to check some important information such as the list of staff and the budgets available.

The aim of my internship was the development of a new version of the Purchase Information System integrated in the university's management portal Cockpit. For this purpose, both the layout and the code had to be completely rewritten. The logic of the program remained the same, following specific workflows: for example in the case of a new purchase request each person in the workflow should approve the request before it is sent to the next responsible, until it is definitely marked as approved and passed to the person in charge of performing the purchase. Therefore, the part of the logic which was stored in the database, such as the sending of e-mails to the different supervisors, was kept unmodified.

In order to be integrated in Cockpit, the programming language had to be changed from Visual Basic to C#. As several functionalities were not used anymore and the old code was extremely long and complicated, the new C# code was completely written from scratch, utilizing the old code only as a guideline.

The layout had to be adapted to the one of Cockpit, with light green colors and rounded-edges objects instead of dark blue and grey colors, sharp edges and straight lines. It was created using CSS instead of the tables used in the old version: this results in a better control of the layout and will be useful in case of a future redesign of Cockpit, since only the references to the stylesheet will have to be changed. The new interface results as more intuitive and easy to use.

During the development, also the most relevant problems of the application were solved: for instance, the speed of the system was increased and the information is now visualized in the correct language, German or Italian, and not a mixed visualization as before.

The application is now in the testing phase and will be added to Cockpit during the next months, including a widget to reach it from the Cockpit main page.

## 1. Introduction

The Free University of Bozen-Bolzano was founded 17 years ago, in 1997, initially as a very small institution with few faculties and courses. It was, however, expanded in the course of time, reaching in 2014 a total of 352 staff members and 3183 students (Free University of Bolzano-Bozen, Numbers, Facts and Figures, 2014).

During this time, a number of applications were created for the internal use of students, professors and administrative staff, in order to ease the management of data and the information flow between the institution and its users, from updating students' careers and lectures' timetables to controlling university's budgets, and many more.

During the last decade, however, also the technologies available for building such applications have remarkably changed, and often applications developed 10 years ago seem tremendously ancient if used nowadays. This is the case of the university's Purchase Information System (PIS), one of the first applications developed by the university in 2002. It is clearly affected by its age, displaying a quite old interface. In fact, both the layout and the development reflect the style of the years of their creation. For this reason, there was the need to renew the old interface and adapt it to the layout of the university management portal Cockpit, a container that groups all university's functions. Moreover, some functionalities had to be changed as a consequence of changing needs and requirements.

Therefore, this work will deal with the renewals and improvements brought to PIS during my internship.

### 1.1 The University and its Need for a Purchase Information System

The university employs 249 administrative and technical staff members in different departments, one of which being the ICT (Free University of Bolzano-Bozen, Numbers, Facts and Figures, 2014).

The Department Information & Communication Technologies (ICT) is responsible for the activities regarding the digital processing of information and communication. It is in charge of maintaining the Scientific Network South Tyrol, organizing the management of the computer network and of all the electronic sequences, developing software to facilitate the workday at the university, and coordinating all tasks concerning the digital processing of information and communication within the Scientific Network South Tyrol (Free University of Bolzano-Bozen, Welcome to the Department for Information and Communication Technology (ICT), 2014).

The ICT department is divided in 4 working groups, which are IT-Management, System Administration, Knowledge Engineering and Software Development. My internship was conducted in the area of software development. Its central scope was the porting of the Purchase Information System (PIS) of the university in the management portal Cockpit.

During time, the main functionalities of the university, such as the student portal or the professor register, were ported in Cockpit. However, this porting was still an open point regarding the Purchase Information System.

The current version of PIS was developed during the first years after the opening of the university and is therefore affected by its age. In fact, both the layout and the implementation have a style that recalls the nineties: buttons and objects inside the page have rectangular shapes with sharp edges, inserted in a dark blue and gray context. The application was developed in Visual Basic and a big part of the application's logic is in the database in PSQL.

Cockpit, instead, has an innovative layout with rounded edges and fresh, greenish colors. It exploits the advantages of the .Net framework, one of the best platforms to develop fully functioning websites, with simplified development and debugging and easy deployment. Cockpit is developed mainly in C#.

The Purchase Information System fulfills many different functions inside the university: firstly, it is used by employers to request the purchase of goods or services, from the purchase of new laptops and books to the request of collaborations/expert advices or business travels.

A second important function of the PIS regards the delegations: when a worker is on holiday, he can delegate a colleague to perform his tasks through the PIS system.

Moreover, PIS can be used to see the list of staff and suppliers with their contact information, to get an overview about the personal situation regarding requests, delegations etc., and to discover all cost centers and budgets related to the person.

### 1.2 Cockpit

Cockpit (Figure 1) represents a central visualization of all applications and processes related to a user account. It is a portal dedicated to students, professors and workers of the Free University of Bolzano, that allows them to keep an eye on the plan of study, exams and grades, list of books and other media borrowed from the Library, expenditures incurred within the University and the Student Card (Mensa, UniBar, photocopies), clocking in/out and many more.

Moreover, thanks to Cockpit news and information regarding the University can be organized in a personalized way, according to the interests of the individual user. Everyone can configure his cockpit, and make sure to display only the information interesting for him. In this way, the flow of information is reduced, eliminating the superfluous. This is obtained through the use of widgets, graphical control elements in the user interface. The user can choose which widgets to display in his Cockpit start page and every widget contains a summary of the information regarding that function. By clicking on the widget, the user can navigate to the related detailed page.

Cockpit was designed and realized between 2012 and 2013, at the beginning only with few functions available and only one person responsible for it. During time, the quantity of functionalities covered by Cockpit hugely increased, and in the same way the number of employees working on the project.

The screenshot shows the Cockpit start page. At the top, there is a navigation bar with the Cockpit logo, a date and time display (tuesday, september 02, 2014), a user greeting (welcome, sonia frei), and navigation tabs for home, workspace, and forum. A search bar is located below the navigation bar. The main content area is divided into several sections:

- ict news:** A large section featuring the 'ct' logo, the date '01.09.2014', and a notification for 'c't Magazin 19/14'.
- weather:** A weather widget for Bozano, Italy, showing 'Right now', 'Today', and 'Tomorrow' forecasts with temperature, wind speed, humidity, and pressure data.
- science south tyrol:** A widget displaying news from 'Tageszeitung' with a list of articles under 'NEWS' and 'EVENTS' sections.
- articles:** A widget titled 'ARTICLES IN JOURNALS' showing a list of articles, including 'UPDATE OF INDICATORS AND MAPS (2011-2014) HARMONISED DATASETS ON LOCAL UNITS (LAU 2) - THE RELEVANCE OF MUNICIPALITY DA...' and 'La mobilità sostenibile tra destination management e mobility management...'.
- collage:** A widget titled 'collage' showing a list of items, including 'Django Reinhardt 05.12.2013' and 'Nachhaltigkeit: Herbsttauschmarkt...'.

Figure 1: Cockpit start page

### 1.3 Contributions

The most manifest change brought to the system during my internship, which is immediately revealed by looking at the new version of Pis, regards the visual part: the layout is in fact considerably different, completely integrate in Cockpit and in line with the current time.

From the point of view of the implementation, the main programming language used was migrated from Visual Basic to C#.

Some functions present in the old version, such as the part regarding the list of possible workflows for a request, were removed, as they were never used and resulted in merely burden the usability of the application.

The functions to extract data from the database were implemented in a class library, which can be utilized in the future for the development of the application for smartphones (e.g. Android) without the need to rewrite them.

## 1.4 Organization of the Thesis

This work will begin with a description of the current state of the Purchase Information System.

First I will focus on the implementation part, more specific on the programming language. The presentation of the interface will follow with some screenshots.


The second part will depict the new implementation of PIS in Cockpit and will be divided in the exposition of the application's logic and the description of the programming choices regarding the programming language and the new interface.

The last part regards the evaluation of the results, including the illustration of the solved problems and the displaying of the new layout.

## 2 Current State of PIS

The current Purchase Information System (Figure 2) is written in Visual Basic and utilizes an Oracle database for the storage of data. The database contains also an important part of the logic in PostgreSQL and connects the different information with each other. The Visual Basic part is extremely huge, including more than 20000 lines of code.

The original developer of the Purchase Information System abandoned the project in 2009. From that moment, the project was committed to another university employer. However, given the enormous size of the code, the new responsible did not implement big changes to the application, since it would have required a complete knowledge of the code. Therefore, only small updates and additions were possible during the last 5 years.


**PIS** 

**REQUESTS**

- New
- Search
- Overview
- Detail
- Invoices

**BASE DATA**

- My data
- Workflows
- Cost centers
- Active budgets
- Staff
- Suppliers
- Logout



# Purchasing Information System

Business processor TEO

**New features in version 3.2.2:**

- New "print" menu exports PIS request to pdf.
- Group "Viaggio di servizio contabilità / Dienstreise Buchhaltung" has been deactivated!  
Use new field "payment method" in position table instead to mark expense positions to be payed directly by accountancy.
- New field "status" is shown in the position table.
- Use new position type "Forfait per vitto e alloggio / Pauschale für Verpflegung und Uebernachtung" in order to account expenses by flat rate.
- New fields on position detail site to handle foreign currency conversion.
- New limits for advance position are implemented.

- [Please look at the demos to understand how requests are organized in positions.](#)  
There are predefined position types depending on the selected request group. Especially for business travels, training courses and collaboration the related types have to be used in order to guarantee that your request will be processed easily and quickly.
- Short user introductions, see under central help clicking on above menu symbol "?"

Figure 2: current PIS start page



## 2.1 Programming Language

The previous version of PIS was developed in Visual Basic.

Visual Basic, first released in 1991, is a third-generation event-driven programming language from Microsoft. Its scope was to be relatively easy to learn and use.

The final release was version 6 in 1998 and from that moment Visual Basic 6.0 IDE is unsupported. However, the community of programmers developed new third party components over time, keeping this programming language in line with modern standards. For these reasons, even now an interesting number of old school programmers still choose Visual Basic as a programming language, in spite of its not negligible disadvantages. In fact, the development environment is no longer supported by Microsoft. Users report versioning problems associated with various runtime DLLs, calling them DLL hell (Microsoft, 2013).

## 2.2 Interface

The PIS system was originally developed in 2002, and its layout clearly reflects the period of its creation, with the classical style used in the late nineties.

In August 1991, Tim Berners-Lee published the first website, a simple, text-based page with a few links. Subsequent pages were similar, namely entirely text-based and with a single-column design with inline links.

Prior to the late 90s, high-speed internet connections did not exist. Therefore, websites from that period needed to be built for very slow connection speeds: they were largely comprised of text, and the design layout we are nowadays used to did not exist (Myia, 2013).

By the mid-90s, web design had evolved both in terms of structure and appearance. In fact, although the original table markup in HTML was meant for displaying tabular data, designers quickly realized they could utilize it to give structure to their designs, and create more complicated, multi-column layouts than HTML was originally capable of. Sites were still based primarily on text, but this text could now be divided into columns, rows, and other navigational elements. Graphical design elements also quickly grew in popularity.

This era of web layouts paid little attention to semantics and web accessibility, often opting for aesthetics over good markup structure (Chapman, 2009).

CSS-based designs started gaining in popularity only after the dotcom boom in the early 2000's. Even if CSS had been available long before then, there was limited support for it in major browsers and many designers were unfamiliar with it.

The PIS system does not make use of CSS: its formatting is solely based on html tables. The design reflects the trends of the nineties, which were spread in each aspect of life, from interior design to cars design to web applications: right-angled shapes and straight lines.

## 2.3 Problems

In the course of time, the Purchase Information System revealed some technical problems, which were solved in the new Cockpit integration.

A first issue regarded the languages chosen for the showing of data. Being the Free University of Bolzano a three-lingual university, users can have English, Italian or German mother tongue. However, the current Pis version is not able to distinguish the language of the user, so information is shown in Italian or German regardless of the user's mother tongue. Through the integration in Cockpit, this problem can easily be solved, as Cockpit has a specified language assigned to each user.

The retrieval and elaboration of data is particularly slow in this version. The speed was enhanced in the C# implementation.

Some aspects of the user interface are not intuitive. For instance, when the list of requests made by the user is shown, there is the possibility for the user to see the details for each request. However, the detailed view of the particular request is visualized only by double clicking on the row containing it. This is extremely user-unfriendly, as this option is not suggested in any way.

### 3 Porting the PIS to the Cockpit

My contribution to the Purchase Information System was a complete restyling of the application: in order to integrate the system in Cockpit, not only the layout had to be changed and adjusted to the Cockpit's one, but also the programming language had to be adapted to the one used in Cockpit, namely C#. Initially, the idea was to convert the script from Visual Basic to C# with an automatic tool and work on this converted version of the code. However, this immediately revealed as a too difficult task: in fact, the existing code was too long and complex and many parts were not actually used. Moreover, the code in C# and the parts in html and Javascript had to be changed too much. Therefore, I chose to write the new code from scratch based on the current version of the system and referring to the old code only in some occasions.


The new Purchase Information System was developed using Visual Studio 2012 Ultimate.

All PIS pages have been implemented following the Cockpit standards. The new layout is in line with the one of Cockpit. A comparison example is represented by Figure 3 and Figure 4, which depict the staff page in the old version (Figure 3) and in the new version (Figure 4).
















Last name ▼	First name	Structure	User name
Abate	Michaela Verena	Baur Siegfried	MicAbate@unibz.it
Abraham	John	PHD STUDENTS TN	JAbraham@unibz.it
Abrahamsson	Pekka Kalevi	Abrahamsson Pekka Kalevi	PAbrahamsson@unibz.it
Acciardi	Martina	Secretariat Faculty of Technology and Natural Sciences	MaAcciardi@unibz.it
Accinelli	Cesare	Faculty of Technology and Natural Sciences	CAccinelli@unibz.it
Achmüller	Barbara	PHD STUDENTS BWS	BAchmueller@unibz.it
Adami	Laura	PHD STUDENTS BWS	LAdami@unibz.it
Agostini	Evi	PHD STUDENTS BWS	EAgostini@unibz.it
Al-Shammari	Ali Fawzi Najm	PHD STUDENTS INF	aalshammari@unibz.it
Alber	Lorenz	I&CT	LAlber@unibz.it
Alberti	Luigi	Gasparella Andrea	LAlberti@unibz.it
Alemi	Maurizio	Purchase	MAlemi@unibz.it
Alessandra	Melonio	Nutt Werner	MAlessandra@unibz.it
Ali	Antonino	School of Economics	AAli@unibz.it
Alizadeh	Pegah	Succi Giancarlo	PAlizadeh@unibz.it
Aluffi Pentini	Anna	Faculty of Education	AAluffi-Pentini@unibz.it
Alzate	Juliana	PHD STUDENTS TN	JAlzate@unibz.it
Amalfi	Maria	Calvanese Diego	MAmalfi@unibz.it
Ammendola	Christian	Gamper Johann	ChAmmendola@unibz.it
Amort	Helmut	Careers Advisory Service	HAmort@unibz.it
Amplatz	Kurt	Maintenance and property	KAmplatz@unibz.it
Andergassen	Ute	Legal Office	UAndergassen@unibz.it
Andergassen Solva	Marlies	Secretariat Faculty of Design	MAndergassenSolva@unibz.it
Andreopoulos	Dimitrios	PHD STUDENTS TN	DAndreopoulos@unibz.it
Andreotti	Carlo	Andreotti Carlo	CAndreotti@unibz.it
Andriolo	Alessandro	Faculty of Technology and Natural Sciences	AAndriolo@unibz.it
Andritsos	Periklis	Gamper Johann	PAndritsos@unibz.it
Angeli	Barbara	Purchase	BAngeli@unibz.it
Angeli	Sergio	Angeli Sergio	SAngeli@unibz.it
Angerer	Victoria	Secretariat Faculty of Technology and Natural Sciences	ViAngerer@unibz.it
Angerer	Evelyn	International relations office	EvAngerer@unibz.it
Antonello	Silvana	PHD STUDENTS BWS	SAntonello@unibz.it
Antonello	Andrea	Tonon Giustino	AAntonello@unibz.it
Antonucci	Daniele	PHD STUDENTS TN	DAntonucci@unibz.it
Arambula Lara	Rigoberto	PHD STUDENTS TN	RArambula@unibz.it
Armondini	Lorena	University library BZ	LArmondini@unibz.it
Artale	Alessandro	Artale Alessandro	AArtale@unibz.it
Asper	Claudia	Secretariat Faculty of Computer Science	CAsper@unibz.it
Astromskis	Saulius	PHD STUDENTS INF	SAstromskis@unibz.it
Atzeri	Anna Maria	PHD STUDENTS TN	AAtzeri@unibz.it
Auckenthaler	Adolf	President	AAuckenthaler@unibz.it
Augschöll	Annamarie	Augschöll Annamarie	AAugschoell@unibz.it
Augsten	Nikolaus	Gamper Johann	NAugsten@unibz.it
Averjanova	Olga	Succi Giancarlo	OVerjanova@unibz.it

Figure 3: Old staff page

thursday, september 04, 2014  home workspace forum [input](#) | [settings](#) | [admin](#) | [log off](#)

**Cockpit**

**Menu**

-  News ▶
-  Events ▶
-  Press ▶
-  Lectures ▶
-  Miscellaneous ▶
-  Articles ▶
-  Professors ▶
-  Students ▶
-  E-university ▶
-  Card watch ▶
-  My Management ▶
-  PIS ▼
-  Budgets

User

Last name	First name	Structure	Email
Abate	Michaela Verena	Baur Siegfried	Michaela.Abate@test.it
Abraham	John	PHD STUDENTS TN	John.Abraham@natec.test.it
Abrahamsson	Pekka Kalevi	Abrahamsson Pekka Kalevi	Pekka.Abrahamsson@test.it
Acciardi	Martina	Secretariat Faculty of Technology and Natural Sciences	Martina.Acciardi2@test.it
Accinelli	Cesare	Faculty of Technology and Natural Sciences	Cesare.Accinelli@test.it
Achmüller	Barbara	PHD STUDENTS BWS	Barbara.Achmueller@education.test.it
Adami	Laura	PHD STUDENTS BWS	Laura.Adami@education.test.it
Agostini	Evi	PHD STUDENTS BWS	Evi.Agostini@education.test.it
Alber	Lorenz	I&CT	Lorenz.Alber@test.it
Alberti	Luigi	Gasparella Andrea	Luigi.Alberti@test.it
Alemi	Maurizio	Purchase	Maurizio.Alemi@test.it
Alessandra	Melonio	Nutt Werner	Melonio.Alessandra@test.it
Ali	Antonino	School of Economics	Antonino.Ali@test.it
Alizadeh	Pegah	Succi Giancarlo	Pegah.Alizadeh@test.it
Al-Shammari	Ali Fawzi Najm	PHD STUDENTS INF	alifawzinajm.alshammari@stud-inf.test.it
Aluffi Pentini	Anna	Faculty of Education	Anna.Aluffi-Pentini@test.it
Alzate	Juliana	PHD STUDENTS TN	Juliana.Alzate@natec.test.it
Amalfi	Maria	Calvanese Diego	Maria.Amalfi@stud-inf.test.it
Ammendola	Christian	Gamper Johann	Christian.Ammendola2@test.it
Amort	Helmut	Careers Advisory Service	Helmut.Amort@test.it
Amplatz	Kurt	Maintenance and property	Kurt.Amplatz@test.it
Andergassen Sölva	Marlies	Secretariat Faculty of Design	Marlies.AndergassenSoelva@test.it
Andergassen	Ute	Legal Office	Ute.Andergassen@test.it
Andreopoulos	Dimitrios	PHD STUDENTS TN	Dimitrios.Andreopoulos@natec.test.it
Andreotti	Carlo	Andreotti Carlo	Carlo.Andreotti@test.it
Andriolo	Alessandro	Faculty of Technology and Natural	Alessandro.Andriolo@test.it

Figure 4: New staff page

### 3.1 Logic of the Program

In order to perform all its functions, PIS has a quite complex logic behind, which had already been decided and was reused for the new implementation.

The most important PIS function is the request of purchase of goods and services. In this case, the user has to select a related group of goods to which his request belongs (such as books, furniture, software...) and a budget that will be debited (such as outsourcing expenses, I&CT hardware expenses...). Moreover, a description of the product should be added, with the explanation of the reason for the request. A limit of money for the purchase has to be indicated, which should represent the maximum amount that will be spent. The request is later sent first to the responsible of the group and only in case of his approval will be sent to the responsible of the budget. If the request is approved also by the latter one, it is marked as approved and is forwarded to the person in charge of satisfying it, for example by ordering a new computer. In this process, the budget limits should be taken into account by the different approving persons and in case of approval the amount indicated is reserved to that action and subtracted from the total budget. At the end of the purchase process, the actual expense is inserted and, in case it is less than it was

foreseen, the remaining amount will be added again to the available budget. This process is better described by Figure 5.

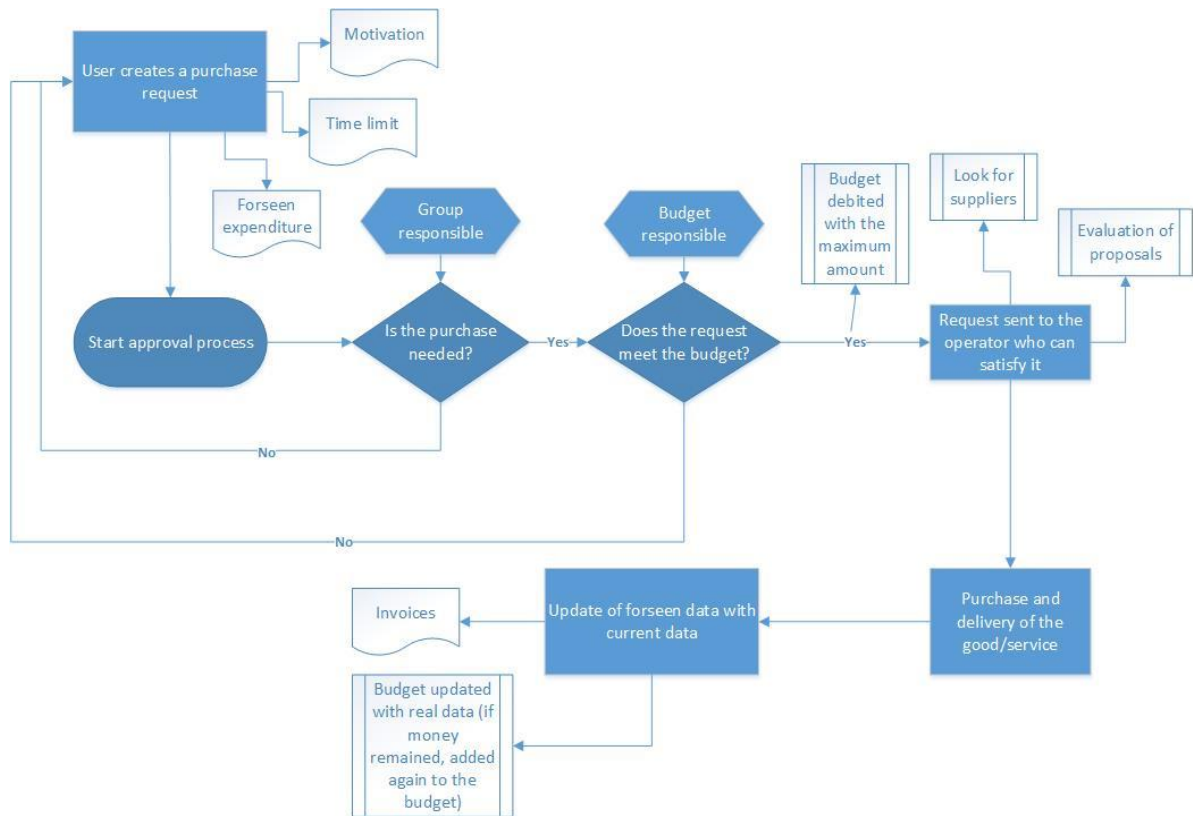


Figure 5: New purchase request logic

The user can later check the status of his requests and a brief description in the overview page, or a detailed description in the detail page. From here, he can also choose to export this information to Excel. Through these pages, the different supervisors can approve or reject the requests, add some comment, change the price limit or the description.

A second important functionality of Pis is the managing of delegations. When going on vacation, an employee may want to give control to his e-mails and approval competences to another person. He can do this in Pis in the My Data page. Here the user can see his own basic information (username, e-mail, employing structure), followed by the list of people who delegated this person, with the relative starting and ending date and the length of the time interval. The second part of the page is dedicated to the delegations: the user can select from a dropdown list the person he wants to delegate and input the dates, or modifying an existing delegation. When confirming the data inserted, the validity of the information is checked: the ending time should be after the starting time, the date should be written in a valid format, there should not be another person already delegated by the same user during that period. In case of correct information, the delegation is saved in the database, or the already stored delegation is updated and the correspondent interval in years, days and hours is calculated.

Moreover, on the Cost Centers page the user can visualize all cost centers related to him and select one of them to view the connected budgets with the time periods in which they are active.

In the Active Budgets page a list of the budgets at disposal for the employee which are currently active is visualized.

In the Staff page a list of all current staff members with the structure they are working in and their email is shown.

The list of all suppliers, instead, can be found in the Suppliers page. When clicking on one entry on the list, complete information about that supplier is visualized, including the address, the e-mail, the phone number and fax number, if it is active at the moment and, if not, the reopening time and, in case, some additional information.

## 3.2 Programming Language

Cockpit is developed in C#: this multi-paradigm programming language was developed by Microsoft within its .NET initiative and encompasses strong typing, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines.

Being a .NET language, C# supports language interoperability, i.e. it can access code written in any .NET compliant language and can also inherit the classes written in these languages. This is not possible in other languages, such as Java (Wadje, 2013).

The environment in which the software to be developed will run has a great importance in the choice of the best language to work with. Windows is the dominating Operating System on client computers. Moreover, the best GUI framework for Windows applications is the .NET Framework and the best programming language to work with the .NET Framework and its APIs is C#.

Therefore, in an environment with Windows clients, Windows servers, Active Directory, IIS and SQL Server such as the Free University of Bolzano C# is the far best language with the .NET Framework (For what reasons should I choose C# over Java and C++, 2011).

## 3.3 Implementation Choices

The web part of the application is developed in html and Javascript. To be integrated in Cockpit, the code calls the Cockpit master page in order to have a uniform layout, with the Cockpit lateral bars which are equal for each page. Through Javascript the web service is called, which requests data from the database. For example, when the user confirms a new purchase request, the web service sends all the information to the Oracle database. The request should then be forwarded to the responsible of the group to give his authorization: this is performed by database scripts in Psql, which send an email to the involved person. In case of an approval, Psql scripts are responsible to check if the foreseen expenditure meets the budget limits. Moreover, also the rest of the approval logic is managed by the database, which is in charge of continuing the purchase

workflow until the request is definitely approved or rejected. At this point, the part regarding the interaction with the user who performed the request is managed by the C# code: it returns the data about the approval/rejection to the web service, which will then display them in Cockpit thanks to Javascript.

### 3.3.1 System Architecture

The application’s workflow starts with the user who selects one of the PIS’ functions. The request is elaborated and forwarded to the web service, which is in charge of contacting the database to obtain the needed information or to save the new information in the database. The database’s response is received by the web service and passed to the web application in order to be displayed for the user, as shown in Figure 6.

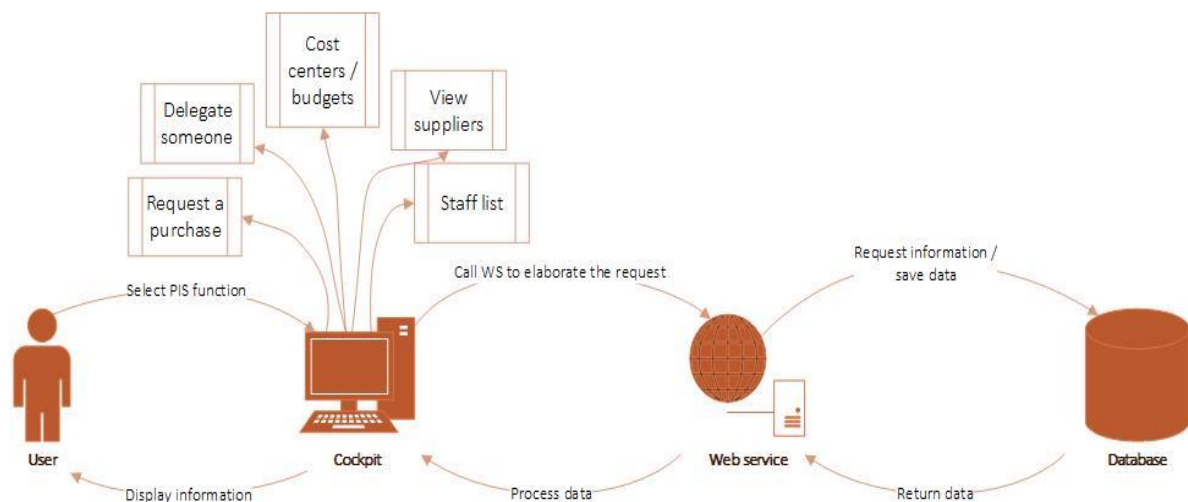


Figure 6: New system architecture

### 3.3.2 The Code

The web part of the application was developed in html using Javascript to allow client-side scripts to interact with the user.

JavaScript (often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, mostly known as the scripting language for Web pages, but used in many non-browser environments as well. It is a prototype-based, multi-paradigm scripting language that is dynamic, and supports object-oriented, imperative, and functional programming styles (Scholz, 2014) (Crockford, 2008).

All data used by the application had to be taken each time from the Oracle database. The programmatic interfaces made available to perform this, that is to communicate between applications, are web services: they are defined by the World Wide Web Consortium (W3C) as “a software system designed to support interoperable machine-to-machine interaction over a network”. A web service “has an interface described in a machine-processable format (specifically

WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP (Simple Object Access Protocol) messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards”.

Web services provide a standard means of interoperating between software applications running on a variety of platforms and frameworks. They are characterized by their great interoperability and extensibility, as well as their machine-processable descriptions, thanks to the use of XML. Web services can be combined in a loosely coupled way to achieve complex operations. Programs providing simple services can interact with each other to deliver sophisticated added-value services (Oracle, 2013).

Therefore, I created a web service to call C# functions from the web application via Ajax calls. Ajax (Asynchronous JavaScript and XML) is a group of interrelated Web development techniques used on the client-side to create asynchronous Web applications. With Ajax, Web applications can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page.

For the purpose of transmitting information between server and web application, Javascript Object Notation (JSON) was used. JSON is a lightweight data interchange format, based on Javascript's object literal notation. Even though it is a subset of Javascript, it is language independent, therefore can be used to exchange data between applications written in all modern languages. Being a text format, it is readable by both humans and machines (Crockford, 2008).

In order to reutilize part of the work done in the future for the development of the same application for smartphones, the functions to extract data from the database were implemented in a class library.

A class library is used in object-oriented programming and is a collection of prewritten classes or coded templates, any of which can be specified and used by a programmer when developing an application program. The programmer specifies which classes are being used and furnishes data that instantiates each class as an object that can be called when the program is executed.

Thanks to class libraries, it results easier to organize and maintain a project. In fact, application extensions can be reused for multiple projects: in this way, code is available that the programmer doesn't have to rewrite. Moreover, a program with a class library runs much faster than one with all the code in the main executable.

### 3.3.3 The Layout

For the look and formatting of the application, the markup language CSS was used: Cascading Style Sheets is a style sheet language used by almost all web pages to describe their presentation.

Styles were added to HTML 4.0 to solve formatting problems. In fact, HTML was never intended to contain tags for formatting a document but rather to define the content of a document, like:

```
<h1>This is a heading</h1>
```

```
<p>This is a paragraph</p>
```



When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers, because the development of large web sites, where fonts and color information were added to every single page, became a long and expensive process. Therefore, to solve this problem, the World Wide Web Consortium (W3C) created CSS: in this way, from HTML 4.0 onwards, all formatting can be removed from the HTML document, and stored in a separate CSS file. Nowadays all browsers support CSS (CSS Introduction, 2013).

One of the most important advantages of CSS is the separation of document content from document presentation, which includes elements such as the layout, colors, and fonts. Thanks to this division, content accessibility is greatly improved and more flexibility in the specification of presentation characteristics is given. In this way, multiple pages can share the same formatting, reducing the complexity and repetition in the structural content. It lightens the code by providing that portion of code that would specify presentation in a separate file.

CSS provides for each relevant HTML element (identified by tags) a list of formatting instructions. For instance, it might give the instruction that all "heading 1" elements should be bold. Therefore, no formatting markup such as bold tags (<b></b>) is needed within the content; what is needed is simply semantic markup saying, "This text is a level 1 heading".

CSS also helps developing a multi platform application: in fact, it allows the same page content to be presented in different styles for different rendering methods, such as on-screen, in print, when read out by a speech-based browser or screen reader and on Braille-based, tactile devices. Thanks to it, the web page can display differently depending on the screen size or on the device on which it is being viewed.

CSS handles the case in which more than one rule matches against a particular element by specifying a priority scheme to determine which style rules to apply. The resulting layout is always predictable, as priorities or weights are calculated and assigned to rules.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318, and they also operate a free CSS validation service (W3C, 2014).

Specifically in the case of Cockpit the use of CSS is very important, because the portal has to maintain a uniform style in all its numerous pages, created by different developers. In this way, once the common layout is defined, each page can refer to the CSS to use the specifications contained in the sheet. Despite the portal is quite new, it has already happened that the layout was completely changed: for instance, some months ago it passed from a blue theme to a green one. Especially in this occasion, having all style information saved in the same file and the different pages referring to that file allowed to completely change the layout without having to modify the code in each page.

The old PIS application layout was styled using html tables. In HTML, tables are defined with the <table> tag. A table is divided into rows with the <tr> tag (tr stands for table row) and a row is divided into data cells with the <td> tag (td stands for table data) (w3schools, 2013).

In the past, tables were used for all sorts of tasks. The most obvious scope is to display data in a table format. However, this was not the sole use: in fact, tabular arrangements were also utilized to align the different elements in a page in a predictable manner. This second use is problematic

because it confuses some software such as screen readers and the position of page elements may not be easily predictable. Moreover, tables are often quite inflexible to work with.

Using the old HTML tags to create tables can be error prone and difficult. Even the self-proclaimed HTML Terrorist David Siegel, the first inventor of the use of tables to contain pure narrative text to format the page, declared in his article “The web is ruined and I ruined it” that tables were not meant for layout scopes and it was a mistake using them in that way (Siegel, 1997).

A new way to handle this matter is by using `<div>` tags: in this way, it is possible to create tables or to logically arrange elements with equal ease without confusing screen readers and other software (Müller, 2014).

The `<div>` tag defines a division or a section in an HTML document and is used to group block-elements to format them with CSS. A `<div>` element can be nested inside another `<div>` element to create different columns of content. Divs are supported by all major browsers: Chrome, Internet Explorer, Mozilla Firefox, Safari, Opera (HTML `<div>` tag, 2012).

Using divs instead of tables to format the page has many advantages.

Firstly, tables are slower. Even if we usually have powerful computers at home nowadays and this problem can seem unimportant, people are using more and more mobile phones to view pages, and they are not as powerful. Divs require less code, which means having smaller files and therefore faster load times.

Moreover, tables are not flexible. In fact, moving an element from one part of the page to another inside tables can be extremely difficult, especially by using Javascript or other scripting languages.

In addition, search engines can fail to understand the relationship from one table cell/column/row to another, as they do not follow how a page looks on the screen but only the outline of the document with headings, sections and paragraphs.

I therefore decided to discard the use of tables to format my application and use divs instead. The understanding of their functionality is slightly more difficult than the utilization of tables at the beginning, but it is soon rewarded by a more fix and easily changeable design of the page.

## 4 Discussion

### 4.1 Lessons Learned

During the internship project, I was able to apply many notions from my study career to a practical project. I already developed many small applications as projects for different courses, but this was the first project of an important size. This helped me understand the importance of a good planning before starting the coding phase and of clarifying with the customers their exact needs before starting the implementation, in order not to have to redesign the whole application.

I deepened my knowledge about databases and SQL and the risks connected to the handling of sensitive data.

Another important lesson I learnt regards collaboration with colleagues and customers: during the university projects, our co-workers are always our colleagues/friends, while the working environment is something rather different. A worker may have to collaborate with someone he never saw before and this collaboration should be optimal even if the two are not close-knit. I had to work with people of different ages, mother tongues, cultures, with different interests and it taught me much from a personal perspective. I even had to collaborate with people working in other countries who I could not personally meet, so we had to talk by online chat.

In addition, having to work with different people underlined the importance of writing good commented code: in fact, as plain English is easier to read than code, comments ease comprehension and can describe things that cannot otherwise be clearly expressed in the source language. If different persons have to work on the same code, it is much easier to understand the parts written by the others if they are good commented. Moreover, even the same person who wrote the code can benefit from good comments in the future, when something needs to be changed in the application or by reusing some parts of it.

Furthermore, I discovered the issues of migrating an application from a programming language to another. Users were already accustomed to use the old application and the major part of them wanted a new version which exactly reflects the old one, even in the not-so-intuitive but at least familiar, parts. It is difficult to satisfy both the old users, who look for something familiar, and the new ones, who look for something more innovative and efficient.

To develop an application starting from an existing code is certainly easier in some aspects, as the idea of how the result should be is already clear. However, there are some disadvantages. Firstly, if the old developer is not available, it can be extremely complicated to understand the code, and the customers are usually not particularly willing to explain again their needs to a new developer. This means that the new developer has to base his comprehension of the requirements completely on the code, which often leads to a repetition of the same suboptimal routines present in the old version.

## 4.2 Advantages of the New System

The new PIS layout follows the one used for Cockpit. It was created using css: in this way, in case of future changes in the layout of Cockpit it will be sufficient to change the css file. The new programming language is C#, so that the application is completely integrated with Cockpit.

The new interface results as more intuitive and easy to use. Other main problems of the system, such as the visualization of the information in the correct language and the slowness of the system, were solved.

Moreover, smaller problems which emerged during the implementation were also solved. For instance, in the cost center page a list of all cost centers is visualized. By clicking on one element of the list, the list of budgets related to it is displayed in another table on the lower part of the page. However, there is no indication of which cost center these budgets relate to. As shown in Figure 7, it is not possible to understand that the user clicked on “telefono” to see the budgets in the lower table.

The screenshot displays the PIS system interface. On the left is a navigation menu with sections: REQUESTS (New, Search, Overview, Detail, Invoices), BASE DATA (My data, Workflows, Cost centers, Active budgets, Staff, Suppliers), and Logout. The main content area is divided into two tables. The top table lists cost centers with columns for Code, Cost center description, and Structure dependent. The bottom table lists budgets with columns for Active from, until, active, Planned, Spent, Left, Budget description, and Budget year.

Code	Cost center description	Structure dependent
AM0A0E	Consulenti/Outsourcing I&CT	Yes
AM0B0E	Students' job I&CT	Yes
CI0000	Software	Yes
LO1000	I&CT	Yes
SI1000	Hardware & Software ICT	Yes
SI1000a	Hardware & Software Fakultäten und Dienststellen	Yes
SI1001	Spese generali I&CT	Yes
SI4013	telefono	Yes

8 cost centers

Active from	until	active	Planned	Spent	Left	Budget description	Budget year
01.01.13	07.03.14	No					2013
01.01.14	31.12.14	Yes					2014

2 budgets

Figure 7: Old cost center page with budgets

In the new version I decided to change the color of the selected budget, therefore from the normal list with all green budgets, the element becomes blue when selected (Figure 8).

Code	Cost Center Description	Structure dependent
CI0000	Software	Yes
SI1000	Hardware & Software ICT	Yes
AM0A0E	Berater/Outsourcing I&CT	Yes
LO1000	I&CT	Yes
SI1001	Allgemeine Ausgaben I&CT	Yes
AM0B0E	Students' job I&CT	Yes
SI4013	Telefon	Yes
SI1000a	Hardware & Software Fakultäten und Dienststellen	Yes
8 cost centers		

Active from	Active until	Active	Planned	Spent	Left	Budget description	Budget year
01-GEN-13	07-MAR-14	No					2013
01-GEN-14	31-DIC-14	Yes					2014
2 budgets							

Figure 8: New cost centers page with budgets

The speed of the application was notably increased: in the old version there was always a gap of about 2 seconds between the time the user clicked on the page to be shown and the visualization of it, reaching a waiting time of 5 seconds in the case of the “my Data” page. In the new version, instead, the page appears after less than 1 second.

The weight of the program was also sensibly reduced: in fact, the old version had about 20.000 lines of code, which were more than halved (to about 7.000 locs) in the new version. The bulky size of the previous version was due to a massive amount of code repetition. Moreover, it contained some unneeded parts, which were removed in the new one.

### 4.3 State of the New PIS

For the time being, all pages were developed separately and a menu to navigate between them is currently missing. The proposal for the complete integration of the application in Cockpit and the navigation between its pages is the creation of a dedicated tab in the upper part of the page, under “Management” (Figure 9). A lateral bar which remains fixed for every page of the application will be shown on the left side. Thanks to this bar, the user can select in which page to navigate. Pages are divided in two main categories “Orders” and “Base data”. The detailed menu

for each category of functionalities will be shown depending on which main functionality the user is using. Orders contains the subpages Overview, New request, Search, Details and Invoices, while Base data contains My data, Cost Centers, Active Budgets, Staff and Suppliers.

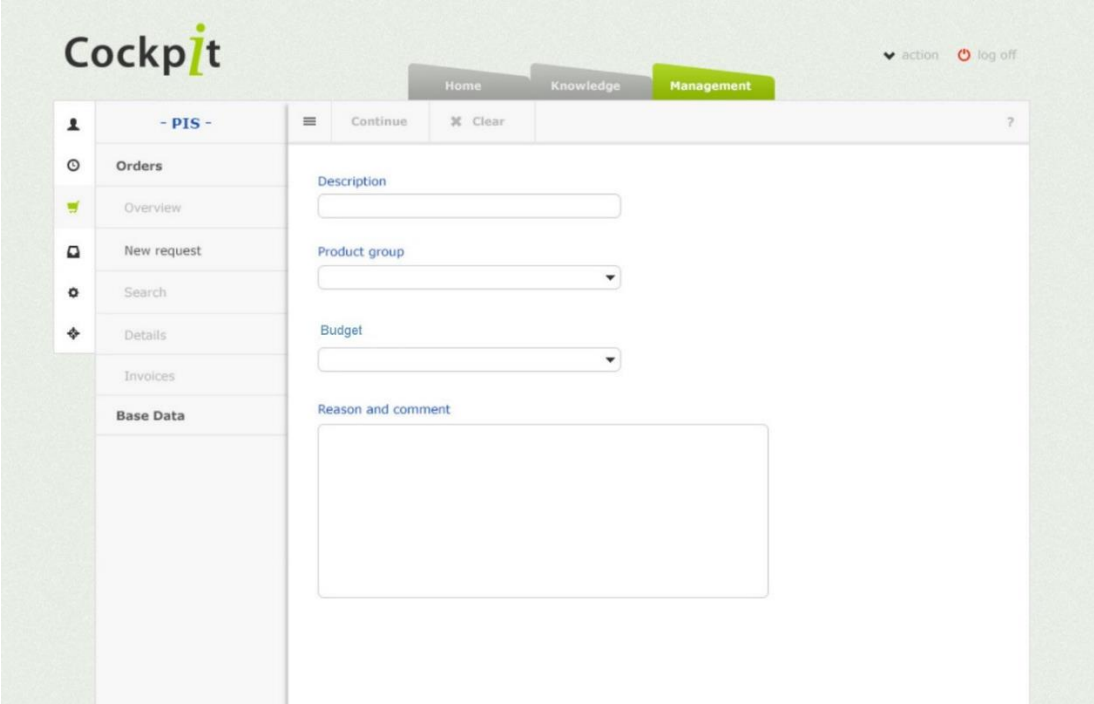


Figure 9: Pis new menu

The application is now in the testing phase and will be integrated in the production Cockpit during the next months. The old version will then be removed. A link to the new version will be added also to the lower Cockpit menu (Figure 10).



Figure 10: Pis link

Moreover, a widget to reach the application from the start page will be linked to the page (Figure 11). As the other Cockpit's widgets, the user can choose to visualize it or not and where to put it inside his start page. The widget summarizes the main PIS information and the user can click on it to reach the Pis page.

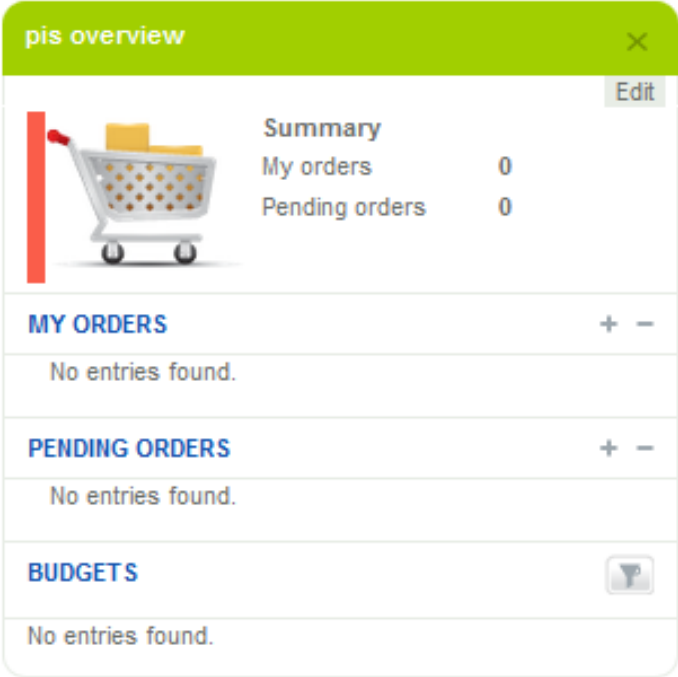


Figure 11: Pis widget

## 5 Conclusion

During my internship, I had the opportunity of applying the theoretical knowledge assimilated during the study to a concrete project. It was extremely useful, as I was assigned a quite important task and had the possibility to learn many new notions in my field. Working in a quite large group, I was surrounded by professionals to whom I could always ask for help and from whom I could absorb new knowledge.

All functions of the old PIS application were implemented in the new version, with the exception of the page Workflows, which was merely a list of supervisors for each group and was not used.

The layout was adapted to the one of Cockpit: therefore, every page looks completely integrated in the portal. In case of a future change in the layout, it will not be necessary to modify the code because the layout is structured with css, so that only the style references will have to be changed.

The functions to extract the correct information from the database are stored in a class library: in this way, when the same application will have to be implemented for smartphones or tablets the same library can be reused.

The PIS system has now to be accurately tested in all its details. Only after this, it will be published and will substitute the old system.

A further addition that would be useful to implement in the near future is the English language support: in fact, as data were stored in the database only in a German and an Italian version, information in English language could not be visualized. Therefore, it would be extremely important to add all specification also in English, in order to fulfill the needs of the increasingly international university staff.



## 6 Acknowledgments

I am heartily thankful to my supervisor, Professor Johann Gamper, who helped me in the development of this work by guiding me and pointing to the most important topics.

I would also like to thank my internship supervisor, doctor Andreas Pircher, who demonstrated a great patience and kindly introduced me to the work environment.

Many thanks to all the university's I&CT staff, especially Manuel Kinzler and Lukasz Karolak, who were always helpful and assisted me in solving every problem I encountered.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project.

## 7 Bibliography

- Chapman, C. (2009, November 28). *The evolution of web design*. Retrieved from Six Revisions: [http://sixrevisions.com/web\\_design/the-evolution-of-web-design/](http://sixrevisions.com/web_design/the-evolution-of-web-design/)
- Crockford, D. (2008). *Javascript: the Good Parts*. Sebastopol (CA): O' Reilly.
- CSS Introduction*. (2013). Retrieved from w3schools: [http://www.w3schools.com/css/css\\_intro.asp](http://www.w3schools.com/css/css_intro.asp)
- For what reasons should I choose C# over Java and C++*. (2011). Retrieved from Programmers StackExchange: <http://programmers.stackexchange.com/questions/125712/for-what-reasons-should-i-choose-c-over-java-and-c>
- Free University of Bolzano-Bozen. (2014). *Numbers, Facts and Figures*. Retrieved from The statistics of unibz: [http://www.unibz.it/SiteCollectionDocuments/2014\\_Numbers\\_Facts\\_Figures.pdf](http://www.unibz.it/SiteCollectionDocuments/2014_Numbers_Facts_Figures.pdf)
- Free University of Bolzano-Bozen. (2014). *Welcome to the Department for Information and Communication Technology (ICT)*. Retrieved from <http://www.unibz.it/en/ict/about/default.html>
- HTML <div> tag*. (2012). Retrieved from w3schools: [http://www.w3schools.com/tags/tag\\_div.asp](http://www.w3schools.com/tags/tag_div.asp)
- Microsoft. (2013). *Visual Basic Language Reference*. Retrieved from Microsoft Developer Network: <http://msdn.microsoft.com/en-us/library/sh9ywfdk.aspx>
- Müller, J. P. (2014). *Using the div tag to create tables*. Retrieved from CSS3 for dummies: <http://www.dummies.com/how-to/content/using-the-div-tag-to-create-tables.html>
- Myia, K. (2013, July 10). *A Look Back at 20+ Years of Website Design*. Retrieved from Hubspot: <http://blog.hubspot.com/marketing/look-back-20-years-website-design>
- Oracle. (2013). *What are web services?* Retrieved from The Java EE 6 Tutorial: (<http://docs.oracle.com/javaee/6/tutorial/doc/gijvh.html>)
- Scholz, F. (2014, August 28). *JavaScript*. Retrieved from Mozilla Developer Network: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- Siegel, D. (1997, October 02). *The Web is ruined and I ruined it*. Retrieved from <http://www.xml.com/pub/a/w3j/s1.people.html>
- W3C. (2014, August 27). *What is CSS*. Retrieved from W3C: <http://www.w3.org/Style/CSS/>
- w3schools. (2013). *HTML Tables*. Retrieved from w3schools: [http://www.w3schools.com/html/html\\_tables.asp](http://www.w3schools.com/html/html_tables.asp)
- Wadje, V. (2013, June 11). *Advantages of C# over Java*. Retrieved from C# Corner: <http://www.c-sharpcorner.com/Blogs/11917/advantages-of-C-Sharp-over-java.aspx>)
- Zhao, J. (2010, April). *Why Java sucks and C# rocks*. Retrieved from <http://www.slideshare.net/jeffz/why-java-sucks-and-c-rocks-final>

## 8 Table of figures

Figure 1: Cockpit start page .....	6
Figure 2: current PIS start page.....	8
Figure 3: Old staff page .....	11
Figure 4: New staff page .....	12
Figure 5: New purchase request logic.....	13
Figure 6: New system architecture .....	15
Figure 7: Old cost center page with budgets .....	20
Figure 8: New cost centers page with budgets.....	21
Figure 9: Pis new menu .....	22
Figure 10: Pis link .....	22
Figure 11: Pis widget .....	23