MASTER THESIS

UNIVERSITAT DE LLEIDA

MASTER'S DEGREE IN INFORMATICS ENGINEERING

SEPA IMPLEMENTATION ON ILERNA ONLINE VIRTUAL CAMPUS

Director: FRANCESC GINÉ DE SOLA

Student: LUIZ FERNANDO JACÇÃO

Course 2016-17
# Table of Contents

**Introduction**  
The company  
The special requirements of the company  
Project goals  
Estimated project time  

**State of the art**  
Overview about the current Learning Management Systems in the market  
The solution  
Advantages of this solution  
Disadvantages of this solution  

**Technologies**  
PHP Script Language  
MySQL Database  
Eclipse IDE  
EMF Framework  
Acceleo Model To Text Language  

**Project Implementation**  
Requirements  
The Strategy Work  
The Technical Work  
   Main module  
   Commercial Group  
   Commercial Activities  
   Invoices  
   Finance  
   SEPA document  

**Case Study**  

**Economic Study**  

**Conclusion**  

**Bibliography**  

**Annex**  
ISO 22022 Main SEPA Message Items Description
1. Introduction

In this first chapter, ILERNA Online will be presented, an innovative online education company that has special software development requirements.

1.1. The company

“ILERNA Online is an online-learning and classroom-learning center, located in the city of Lleida, with more than 50 years in the educational sector and more than 15,000 students graduated.” [3]

ILERNA began its trajectory as a school of classroom-learning but in the last 5 years, it has been betting on online-learning and it is having a great expansion in the learning market.

1.2. The special requirements of the company

ILERNA Online needs a customized virtual campus because it is a company that:

- look for innovation, so the virtual campus should have an interface that is like a social network
• is an online-learning education center and needs a virtual campus that is integrated with its current e-commerce
• is a classroom-learning center and needs a virtual campus that generates the Direct Debit payment system, payment method that allow monthly the billing of the classroom courses.
• is in Catalonia and its virtual campus needs to be adapted to the regional laws, for example to the generation of the GuiaMat file.

1.3. Project goals

To resolve the requirement of the Direct Debit payment system, it was proposed the integration of the SEPA standard[1] into the customized virtual campus with the focus of extending the virtual campus into an Enterprise Resource Planning.

The standard to be integrated is the pain.008.001.02 that it is a “message sent by the initiating party to the forwarding agent or creditor agent. It is used to request single or bulk collection(s) of funds from one or various debtor's account(s) for a creditor.”[1]

In order to achieve this main aim, the following sub-aims have been faced:
• Development of an easy system to make the billing of the monthly student quota in blocks by courses.
• Development of an invoice system.
• Development of a financial system.
• Development of a SEPA document generation system.

1.4. Estimated project time

To estimate the project time, a work breakdown structure was used to package the required effort in tasks and subtasks.

The Work Breakdown Structure was organized into four main packages:

• Development: the main package that have the effort to develop a beta version. This package was organized into:
○ Model package: the effort necessary to collect the requirements and modeling this into a class diagram that will be used also to model the database structure.
○ Service package: the effort necessary to develop the services, the core of business logic.
○ View package: the effort necessary to design and build the view that communicate with the service layer.
  ● Testing: this package represents the time required to run the integrated tests in the beta version and mark the errors that must be fixed in the refactoring package.
  ● Refactoring: this package is the estimated time for the refactoring of the beta version for the stable version.
  ● Final testing: this package is the estimated effort to test the stable version of the application.

The Figure 2 shows the Work Breakdown Structure of the work developed throughout this project.

![Figure 2: Work Breakdown Structure](image)

To make a timeline of this project was used a Gantt Chart Table, which defines the order of the task and the main milestones of the project. This is shown in Figure 3.
The total estimated time was 226 hours as shown in the Figure 4:

<table>
<thead>
<tr>
<th>Name</th>
<th>Effort (plan)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>Σ 188 h</td>
</tr>
<tr>
<td>Model</td>
<td>Σ 40 h</td>
</tr>
<tr>
<td>Service</td>
<td>Σ 100 h</td>
</tr>
<tr>
<td>View</td>
<td>Σ 48 h</td>
</tr>
<tr>
<td>Testing</td>
<td>Σ 8 h</td>
</tr>
<tr>
<td>Refactoring</td>
<td>Σ 24 h</td>
</tr>
<tr>
<td>Final testing</td>
<td>Σ 6 h</td>
</tr>
</tbody>
</table>

Figure 4: Chart of the estimated effort throughout the project.
2. State of the art

This chapter shows why this project was done, presents the current state of the market and a new solution that aims to be better than the current solutions.

2.1. Overview about the current Learning Management Systems in the market

There are many Learning Management Systems (LMS) in the market, for example:

- Free and open-sources LMS like Moodle [9], ATutor [10], Eliademy [11] and Sakai [12]. They don’t have any native integration with any e-commerce and has no Direct Debit integration. It can be customized but customize this application to make as a social network has a high cost because it drastically changes the structure of the software.

- The LMS not free like Maestro [13], SoftAula [14], Geenio [15], can be partially customizable and some have an integrated e-commerce, but none of them have Direct Debit integration or the generation of GuiaMat [16] files.

- The implementation of a mixed solution, that is, the use of one software for one function and another software for another function. This is not a good alternative, since it requires more human work, for example, when there is a change in the data of a student, this change must be made in all software multiplying human effort to accomplish a single task.

2.2. The solution

The solution is to create something new in the market, a solution totally customized to the needs of the company, a solution that makes the company dynamic to take the lead in the market. And the implementation of SEPA is part of this solution.

2.2.1. What is the SEPA?

“The single euro payments area (SEPA) harmonises the way cashless euro payments are made across Europe. It allows European consumers, businesses and public administrations to make and receive the following types of transactions under the same basic conditions
● credit transfers
● direct debit payments
● card payments

This makes all cross-border electronic payments in euro as easy as domestic payments.” [2]

2.3. Advantages of this solution

The main advantages of this solution are:

● A virtual campus adapted to the company's differential idea of an interface like a social network.
● Use of the current e-commerce:
  ○ With a customized solution the current e-commerce can be integrated into the virtual campus.
  ○ With another solution, that have a built-in e-commerce, all the interface customization and customer data of the current e-commerce would have to be migrated.
● Integration of SEPA Direct Debit: ILERNA Online has the dual profile of classroom-learning and online-learning, SEPA Direct Debit allows the billing of the classroom-courses.
● Adaptation to regional laws of education: ILERNA Online is located in Catalonia, region of Spain that has its own educational system, and thus ILERNA Online must provide information using the official structure to the regional government. The customized solution allows the generation of this information with official structure.

2.4. Disadvantages of this solution

The main disadvantage of this solution is the elucidation of a completely new environment, the requirements to be discovered and the solutions to be proven. The LMS tools on the market already have an experience with e-commerce integration, and the Direct Debit tools already have a great experience with SEPA, so the biggest disadvantage in a new research is that they always encounter great difficulties and hidden costs until they reach a stable product.
3. Technologies

This chapter describes the selected technologies used throughout this project.

3.1. PHP Script Language

“PHP, which stands for "PHP: Hypertext Preprocessor" is a widely-used Open Source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. Its syntax draws upon C, Java, and Perl, and it is easy to learn. The main goal of the language is to allow web developers to write dynamically generated webpages quickly, but you can do much more with PHP.”[4]

The main reasons for the selection of this technology are:
- PHP is a script language that allows rapid development because it does not need to be compiled to see the result.
- It has a lot of libraries and frameworks.
- This is a highly accepted language for server environments in the cloud.

3.2. MySQL Database

“MySQL is the world’s most popular open source database. With its proven performance, reliability and ease-of-use, MySQL has become the leading
database choice for web-based applications, used by high profile web properties including Facebook, Twitter, YouTube, Yahoo! and many more.”[5]

The main reasons for the selection of MySQL are:
- It is a free and open-source technology.
- This is a highly accepted Data Base Management System for server environments in the cloud.

3.3. Eclipse IDE

“Eclipse is famous Java Integrated Development Environment (IDE), but the C/C++ IDE and PHP IDE too. You can easily combine language support and other features into any of default packages, and the Eclipse Marketplace allows for virtually unlimited customization and extension.”[6]

Eclipse has many plugins and frameworks that is helpful in the development environment. Basically Eclipse was chosen because:
- It’s free that reduce the cost of development
- Has many plugins like GIT, SASS and JavaScript editor that decrease the effort of development.
- Has the PHP language support, with auto-complete, syntax highlight, wizards and debug.
3.4. EMF Framework

“The Eclipse Modeling Framework (EMF) project is a modeling framework and code generation facility for building tools and other applications based on a structured data model. From a model specification described in XMI, EMF provides tools and runtime support to produce a set of Java classes for the model, along with a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor.”[7]

Eclipse Modeling Framework is a part of the Eclipse IDE. This framework allows the design of all model structure of the application that improve the time of development and maintenance because the model already is a document for the developers use.
The edition of the model is direct in the IDE and can be versioned by GIT technology.

3.5. Acceleo Model To Text Language

"Acceleo is a pragmatic implementation of the Object Management Group (OMG) Meta Object Facility (MOF) Model to Text Language (MTL) standard. You do not need to be an expert to start using the plug-ins and create your first code generator : using the provided example projects and the powerful completion feature of the Acceleo editor, it is very easy to get started and understand the basic principles.

Acceleo is the result of several man-years of R&D started in the French company Obeo. Junction between the OMG MTL standard, its team’s experience with industrial code generation and the latest research advances into the M2T field, it offers outstanding advantages : High ability to customize, Interoperability, Easy kick off, and much more!"[8]
Acceleo is a preprocessor language that converts models into source files. In this project, Acceleo converts EMF models to PHP source files. The developer configures the source model and the destination folder. Using the template files (MTL), the Acceleo preprocessor generates the files for the application. Many sources files can be generated in seconds.

Figure 7: The Eclipse IDE with the Model To Text language

The most useful feature of Acceleo in this project is to convert all models of the database diagram into objects inside of the application and then convert those objects into a schema inside of the database automatically, thus maintaining an integrity between documentation, application and database structure.
4. Project Implementation

This chapter shows the requirements of the project, the strategic plan for the solution of these requirements and the technical execution of this solution.

4.1. Requirements

The basic requirement to be reached is the billing through the Direct Debit from the students of the classroom-learning modality, but this requirement generates other requirements that generate sub-requirements:

- To implement the method of payment by Direct Debit, it is necessary to implement the European standard SEPA pain.008.001.02.
- To implement the SEPA standard, it is necessary to have an invoice system that can:
  - Create invoices
  - Cancel invoices
  - Change invoices
  - Print invoices
- To implement the invoice system, it is necessary to implement also a trade activity system that can:
  - Create multiples activities
  - Cancel activities
  - Change activities
  - Generate invoices from activities
- To generate multiple activities massively, it is necessary the grouping of the students organized in academic courses. That includes:
  - Create a group
  - Destroy a group
  - Change a group
  - Automatically create a group for each academic course by academic session and put the correspondents students in these groups.
4.2. The Strategy Work

The first part of the plan was to define the basic technologies that will be used in the project, which are described in Chapter 3. These technologies were selected based on the time and cost of development. Many technologies, frameworks and libraries was tested and discarded. Suggestions and experiences of others professionals was considered in the selection of the technologies to be used.

The second part is the creation of a test module that allows testing the selected technologies. Many changes was made in the library project until reaches the basic functionality of the module.

In sequence, it was necessary collect all detailed requirements that include many meetings with the product owner and the future system users. After that, we check these requirements with the set of technologies. The requirements was collected with a system user that has experiences on an old system of Direct Debit of SoftAula. In order to do that, we made made screen-shots and appointments. Some of them are shown in the Figure 8.
For better organization, the finance module was divided into the following four submodules:

- **Commercial group**
  This submodule groups by academic program all the students that are coursing the current formative period. When a student enrolls in the campus, this automatically inserts the student in the group of the corresponding course.

- **Commercial activities**
  This submodule lists all business activities and generates invoices for students; as the activities are volatile during the formative period only the concrete activities generate invoices.

- **Finance**
  The Finance submodule shows all financial transactions of the company, and allows the management of these financial transactions.

- **Invoices**
  The invoice submodule allows invoice management in addition to allow the invoices to be printed.

- **SEPA Document**
  The SEPA sub-module generates the SEPA file for a given period of charges by Direct Debit payment method that has invoices, a file that is sent through the web interface of the correspondent bank of the issuer.

4.3. The Technical Work

This chapter describes all the technical work necessary to achieve the goals presented previously. The first step was to develop a base technology that would support the collected requirements, for which a test module was developed, in which the selected technologies and the base technology used by the campus were tested.
4.3.1. Main module

A main module was developed that allows access to the others submodules. This module controls the access permission of the user to the submodule.

4.3.1.1. View

This module consists of only one view, which is shown in the following Figure 9.

![Figure 9: View of the Business main module](image)

This view allows access to the following submodules:

- **Commercial Group**: the submodule that manages student groups. Internally in this project is called "Trade Group", but to make the term clearer, it is called "Commercial Group" in this document.
- **Commercial Activities**: the submodule that manages student groups. Similarly to "Trade Group", the "Trading" is called "Commercial Activities" in this document also to be more clear.
- **Bank**: the submodule that manages the bank registers.
- **Invoice**: the submodule that manages the invoices.
- **SEPA**: the submodule that generates the SEPA documents.
- Finance: internally in this project, it is called “Enterprise Account. This is the submodule that manages the finances grouped by the accounts of the company.

4.3.2. Commercial Group

The Commercial Group was the first submodule to be developed because of the dependence of the other submodules by it. In all modules, the first task performed was the model of the database, through the requirements collected.

4.3.2.1. Model

The following diagram, in the Figure 10, represents the Commercial Group submodule:

![Diagram](image)

Figure 10: Commercial Group Submodule Diagram.

The entities are modeled according to the requirements collected:
• Person entity: represents a person who is extended to a PersonLegal or a PersonNatural. This model is to satisfy the requirement of Debit Direct billing of individual and legal entities, because exists the possibility that is not the student who pays for the course.
• PersonLegal: represents a company, a partner or an education center
• PersonNatural: represents a student, a teacher or a system user.
• PersonAddress: This entity represents the 1: N connection between the Person entity and the Address entity, because a person can have more than one address, for example an invoice address and a billing address.
• FinancialAccount: This entity represents the accounts that a person can have. This entity allows the financial movement between the accounts.
• FinancialAccountBank: This entity represents the bank account, an extension of the "FinancialAccount" entity.
• Bank: an entity that represents the bank of the FinancialAccountBank entity, the SEPA standard requires the bank's BIC code.
• CompanyAddress: entity that represents the different addresses of a company, because each address of an education center has its own group of students. For example: ILERNA Lleida, ILERNA Madrid and ILERNA Italia.
• TradeGroup: entity that represents a group of students that is associated with a company address.
• TradeGroupPerson: entity that represents the student and their address on the invoice, the payer's account and the group that the student belongs to.

4.3.2.2. Service

TradeGroupService is the main service of this module, it has CRUD functions that allow you to create, read, update and destroy records of the TradeGroup entity.

The "generateTradingFromObj" function is responsible for creating a commercial activities through the wizard. This method has the following parameters:
• The "TradeGroupPerson" that indicates to whom will be generated the commercial activities.
• The object that contains the generation date, the address of the company that will receive the activity and the initial state of this activity.
• The items and the prices of this activity
• Payment dates and amounts
4.3.2.3. View

The view of the Commercial Group, shown in the Figure 11, contains:

- The group code, the campus automatically fills this field with the course code for the group.
- The title of the group, the campus automatically fills this field with the title of the course for the group.
- The company address which the group belongs.
- The list of student and its payers that belongs to the group.

![Image of the Commercial Group submodule](image.png)

Figure 11: View of the Commercial Group submodule.

4.3.3. Commercial Activities

The second submodule developed was the "Commercial Activities", these activities are generated from the "Commercial Group" submodule.

4.3.3.1. Model

Similarly to the other modules, the entities are modeled according to the requirements collected and are related as shown in Figure 12:
The entities are:

- **Trading**: the main entity of this submodule and has these main fields:
  - `sequenceNumber`: auto-increment number associated to one record.
  - `tradingDate`: the date that the activity occurred.
  - `tradingCompanyAddress`: the address of the company where the activity occurred.
  - `personAddress`: the address of the student that will be indicated on the invoice.

- **TradingGroupPerson**: the connection N:N between Trading and TradeGroupPerson, that indicate which group this activity came from.

- **TradingOperation**: the operation of the activity, at this moment only the "Course activities" exists, but can be any type of operation of the company, for example, "Sale of academic material" or "External activities".

- **TradingStatus**: the status of the activity, for example “Open”, “Close”, “Cancelled”, etc…
• *TradingItem*: the elements that compose the activity, for example "Academic Quota", "Book", "Certificate of completion of the course", etc ...
• *TradingHistory*: the status history of the activity. Is only used to audit the activity in case of human error.
• *FinancialTransactionTrading*: the association of the activity with the financial payments.

4.3.3.2. Service

The *TradingService* is the main service of this submodule, like the others services has the CRUD service for the entity Trading. The method *generateInvoiceFromTrading* is responsible to generate the Invoice from the "Commercial activities", and expects these parameters:

• *Trading*: the activity that generates the Invoice
• *TradingStatus*: the new status of this activity after the successful generation

4.3.3.3. View

The view of the Commercial Activities is shown in the Figure 13.

![Figure 13: The view of the Commercial Activities submodule.](image)
The view of the Commercial Activities contains:

- The number sequence of the activity, this is an auto-increment number
- The date of the activity generation
- The status of the operation
- The person associated with this activity
- The payment preview date
- The amount to be paid
- The transaction method
- The current status of the financial transaction

4.3.4. Invoices

The third submodule developed was the "Invoices", these invoices are generated from the "Commercial Activities" submodule.

4.3.4.1. Model

The Figure 14 represents the Invoice submodule, which has the following entities:

- Invoice: represents the invoice record and has the following fields:
  - invoiceSeriesCode: the prefix of the invoice. Is usually related to the address and the company that generate the invoice.
  - invoiceNumber: the number of the invoice, auto-increment by series code.
  - status: the status of the invoice, which can be: “Printed”, “Cancelled” and “SEPA generated”.
  - invoiceClass: indicates whether the invoice is “original” or “corrective”. It is a preparation for the Spanish electronic invoice.
- InvoiceItem: the connection between the invoice and the items of the activity.
- InvoiceTransaction: the connection between the invoice and the financial transactions of the activity.
4.3.4.2. Service

The InvoiceService is the main service of this submodule, like the others services has the CRUD service for the entity Invoice. The printInvoice method is responsible for printing the invoice and expects the invoice_id as parameter. The printed invoice result is shown in Figure 15.
Figure 15: Example of a printed invoice

The printed invoice has the “Delivery address” and the “Billing address”, because the person, who pay, can be distinct of the person that receives the product or service.
4.3.4.3. View

The view of the Invoice submodule is shown in the Figure 16.

![Invoice submodule view](image)

Figure 16: The view of the Invoice submodule.

The view of the Invoice submodule contains:

- The series prefix by company address.
- The number sequence of the invoice, which is an auto-increment number by series prefix.
- The invoice generation date.
- The status of the invoice.
- The person receiving the invoice.
- The invoice items.
- The price of the items.
- The quantity of the items.

4.3.5. Finance

The other developed submodule was the "Finance", these finances are generated from the "Commercial Activities" and validated by the "Invoices".
4.3.5.1. Model

The Figure 17 represents the Finance submodule, which has the following entities:

- **FinancialTransaction**: represents the finance record and has the following fields:
  - amount: the value of the transaction.
  - paymentDate: the date of payment.
  - financialFlow: indicates if is cash inflow or outflow.
  - financialTransactionMethod: the payment method.
- **FinancialAccount**: the personal account associated with the transaction.
- **TradeCompanyAccount**: the company account that is associated with the transaction.
- **FinancialCategory**: the category to classify the transaction, useful for generating reports.

![Figure 17: Model diagram of Finance submodule.](image-url)
4.3.5.2. Service

The Finance submodule has no special method implemented because it is a submodule that will be expanded. This submodule has only CRUD operations.

4.3.5.3. View

The view of the Finance submodule, which is shown in the Figure 18, contains:

- The account that is associated with these financial transactions.
- The company that belongs to the account, because the system is multi-company.
- The list of transactions associated with the selected account, each transaction has:
  - Person associated with the transaction.
  - The transaction date,
  - The payment method of the transaction.
  - An auto-increment number associated with the transaction.
  - A text reference for the transaction.

Figure 18: View of Finance submodule.
4.3.6. SEPA document

The "SEPADocument" is the last developed submodule. The SEPA document is generated from "Finance", which has an associated invoice and it has the Debit Direct as payment method.

4.3.6.1. Model

The Figure 19 represents the SEPA submodule, which has the following entities:

- **SEPADocument**: represents a SEPA document.
- **SEPPaymentInformation**: a SEPA document can have a lot of payment information, which can be "Credit transfer", "Direct debit" and "Card payments". This entity is modeled only with the "Debit direct" payment information.
- **SEPATransactionInformation**: the unitary information about a financial transaction, a **SEPPaymentInformation** can have many transactions.

Detailed information on the fields can be found in Annex 9.1

![Figure 19: Model diagram of the SEPA submodule.](image-url)
4.3.6.2. Service

The SEPADocumentService is the main service of this submodule, like the other services, it also has the CRUD service for the SEPADocument entity. The generateSepaDocumentFromInvoices method is responsible for generating the SEPA document for the invoices and expects these parameters:

- Invoice Transactions: the list of invoice transactions that have Direct Debit as payment method.
- Trade Company Account: the account of the company that receives all the transactions.
- Trade Company Address: the address of the company that is creating the document.
- Requested Collection Date: the date for billing.

ISO 22022 is a universal standard and it is adopted by the Single Euro Payments Area as the standard of communication between European financial institutions. But it is not a specific standard for SEPA, so there are generic fields in ISO 22022 that are better explained in other documents. This makes the elucidation of requirements even more complex because there is no single document to be consulted. Examples of these fields are:

- **PrivatIdentifstion** <PrvtId>
  This field is an identification of the issuer of the SEPA document. This identification is provided by the bank after registration and approval by the bank. This information was found with the help of the bank's technical support.

- **PaymentIdentifstion** <PmtId>
  This is a mandatory field of end-to-end unique transaction identifier. There was no way to fill this field because there is no agreed number between students and the educational center for each transaction. The solution was found on a discussion page of the SEPA rules, which informs the "NOTPROVIDED" as a possible value.
4.3.6.3. View

The view of the SEPA document submodule, which is shown in the Figure 20, contains:

- The date that the document was generated.
- The current state of the document.
- The unique message identifier of the document.
- The unique payment identifier.
- The list of transactions associated with the selected SEPA document, each transaction has:
  - Person associated with the financial transaction
  - The amount of the transaction
  - The IBAN account number
  - The invoice number associated to the transaction
  - The unique identifier of the transaction

![Figure 20: View of SEPA submodule.](image)

All fields are read-only because, once the document is generated, the data can no longer be modified.
5. Case Study

This chapter presents a case study of the generation of the monthly quota in the amount of 175.00€ generated for a group of students of the "Nursing Assistant" course.

5.1. Academic activities generation

The first step is trigger, in the window menu, the wizard to generate the commercial activities. The wizard is shown in the Figure 21.

![Figure 21: Form of commercial activity wizard generator](source)

The second step is to fill the form that has:

- The items of the activities with their description and values.
- The company that will generate the activities, because it can be another company or the same company but at another address.
- The initial status of the activities.
- The periodicity, because activities can be generated repeatedly in a period.
- The date of the first activity.
In the next step, it is selected the group of students who will be generated the activities, in this case, we selected the course of "Nursing Assistant".

![Figure 22: Group selection wizard](image)

The next step is to select the people to whom the activities will be generated, because it may be that for some reason a person is not included in this activity.

![Figure 23: Person selection wizard](image)
5.2. Invoices generation

To generate the invoice, the first step is to launch the wizard in the Commercial Activity menu. The wizard is shown in Figure 24.

![Figure 24: Form of invoice wizard](image)

After completing the invoice form, the next step is to select the activities in the date range that the invoices will be generated and click "Generate" to finish.

![Figure 25: Activity selection wizard](image)
5.3. SEPA document generation

The last wizard is the SEPA generation, as shown in Figure 26.

![Figure 26: Form of SEPA wizard](image)

After the company date and company account has been selected, a list of invoices that can be generated in the SEPA document is displayed, as shown in Figure 27.

![Figure 27: Invoice selection wizard](image)
Only invoices that have a "Direct Debit" as your payment method are shown. Only these financial transactions are included in the SEPA document. After that the SEPA XML document is generated, the file is ready to be sent to the bank.

5.4. Sending to the bank

On the website of the company bank, in this case the BBVA, there is an area to test the SEPA files. The result is shown in Figure 28.

![Figure 28: The successful test of the SEPA document](image)

The file has been successfully tested and submitted to the bank. At the specified date, the collection was performed and all Direct Debits received.

This demonstrates that the development work was successful, since this was the main goal of this project module.
6. **Economic Study**

This chapter presents the basic cost with this solution and a simple comparison with another possible solution.

6.1. **Cost of this custom solution**

Based on the total of 226 work hours of this project and a cost of 10.00€ per hour of work for the company, we have approximately a total value of 2,260.00€ of cost for this module.

6.2. **Comparison with other solution**

Using the *SoftAula* solution with its best plan, the "Cloud Ultimate", we have a fixed cost of 125.00€ per month. Comparing with the current solution, we obtain the following graph as shown in Figure 29.

![Cost Comparison Graph](image)

**Figure 29: Cost comparison between solutions**

If we consider the cost of the SoftAula solution as a revenue in this solution, then we have a Break-Even Point (BEP) at 18 months and a Payback Period (PBP) at 3 years.
7. Conclusion

This chapter shows the analysis of objectives, future work and possible improvements.

7.1. Goal Analysis

The main goal was to integrate the SEPA pain.008.001.02 message and, for this, it was necessary to achieve all the sub-aims, which are:

- Development of an easy system to make the billing of the monthly student quota in blocks by courses: a submodule was developed that groups the students by courses and generates the activities for these groups. The technical work of this sub-module is specified in chapters 4.3.2 and 4.3.3 and the test of this sub-module is specified in chapter 5.1. This sub-aim was successfully achieved.

- Development of an invoice system: a sub module was developed that manages the invoice and prints them. The technical work is explained in chapter 4.3.4 and tested in chapter 5.2. This sub-aim has also been fully achieved.

- Development of a financial system: a submodule was developed that manages the finances but only the finances that has the Debit Direct as payment method. A full financial submodel will be developed in a future work. The technical work is shown in chapter 4.3.5. This sub-aim was partially achieved.

- Development of a SEPA document generation system: was developed the SEPA submodule that allow the generation of the SEPA document. The technical work is described in the chapter 4.3.6 and tested in the chapter 5.3. This sub-aim also was fully achieved.

All the sub-aims and the main goal were achieved, a SEPA document was successfully tested in chapter 5.4 and the economic advantages of this solution were demonstrated in chapter 6. This development was only possible by
engineering a solution that fulfilled the requirements of the company in the economic, commercial and technical areas.

7.2. Possible Improvements

To make a rapid development at low cost, a standard interface was used that is not an intuitive interface. A major improvement would be the creation of an intuitive interface using the best User eXperience knowledge.

Another improvement would be the implementation of the Spanish electronic invoice, which allows to commercialize with the public administrations of Spain.

7.3. Future Goals

The future goals of this module are:

● a full financial management system that:
  ○ allows not only the Debit Direct payment method, but also the Credit Transfers and the Card Payment.
  ○ can manage the in and out flow cash of the company.

● a great support for the commercial operations, not only for the collection of the courses, but also the operations like purchases and budgets of the company.
8. Bibliography

September 2009,
https://www.iso20022.org/documents/general/Payments_Maintenance_2009.zip

[2] Single euro payments area (SEPA)
August 2017,

[3] ILERNA Online Website
August 2017,
https://www.ilerna.es/es/

[4] PHP Website
August 2017,
http://php.net/

[5] MySQL Website
September 2017,
https://www.mysql.com

[6] Eclipse IDE Website
September 2017,
https://eclipse.org

September 2017,
https://eclipse.org/modeling/emf/

[8] Acceleo Model to Text language
September 2017,
http://www.acceleo.org/

[9] Moodle LMS Community
September 2017,
https://moodle.org

[10] ATutor LMS Project
September 2017,
http://www.atutor.ca

September 2017,
https://eliademy.com/
[12] Sakai Project  
September 2017,  
https://www.sakaiproject.org/

September 2017,  
https://meetmaestro.com

[14] SoftAula Education Platform  
September 2017,  
http://softaula.net

September 2017,  
https://geen.io

[16] GuiaMat Information Schema  
September 2017,  
9. Annex

9.1. ISO 22022 Main SEPA Message Items Description

The following section identifies the elements of the CustomerDirectDebitInitiationV02 message definition. [1]

1.0 GroupHeader <GrpHdr>
- Presence: [1..1]
- Definition: Set of characteristics shared by all individual transactions included in the message.
- Type: The GroupHeader block is composed of the following GroupHeader39 element(s):
  o 1.1 MessageIdentification
  o 1.2 CreationDateTime
  o 1.3 Authorisation
  o 1.6 NumberOfTransactions
  o 1.7 ControlSum
  o 1.8 InitiatingParty
  o 1.9 ForwardingAgent

1.1 MessageIdentification <MsgId>
- Presence: [1..1]
- Definition: Point to point reference, assigned by the instructing party and sent to the next party in the chain, to unambiguously identify the message.
- Usage: The instructing party has to make sure that MessageIdentification is unique per instructed party for a pre-agreed period.
- Data Type: Max35Text
- Format: maxLength: 35, minLength: 1

1.2 CreationDateTime <CreDtTm>
- Presence: [1..1]
- Definition: Date and time at which a (group of) payment instruction(s) was created by the instructing party.
- Data Type: ISODateTime

1.6 NumberOfTransactions <NbOfTxs>
- Presence: [1..1]
• Definition: Number of individual transactions contained in the message.
• Data Type: Max15NumericText
• Format: [0-9]{1,15}

1.7 ControlSum <CtrlSum>
• Presence: [0..1]
• Definition: Total of all individual amounts included in the message, irrespective of currencies.
• Data Type: DecimalNumber
• Format: fractionDigits: 17, totalDigits: 18

1.8 InitiatingParty <InitgPty>
• Presence: [1..1]
• Definition: Party that initiates the payment.
• Usage: This can either be the creditor or a party that initiates the direct debit on behalf of the creditor.
• Type: This message item is composed of the following PartyIdentification32 element(s):
  ○ 9.1.0 Name
  ○ 9.1.12 Identification

2.0 PaymentInformation <PmtInf>
• Presence: [1..n]
• Definition: Set of characteristics that apply to the credit side of the payment transactions included in the direct debit transaction initiation.
• Type: The PaymentInformation block is composed of the following PaymentInstructionInformation4 element(s):
  ○ 2.1 PaymentInformationIdentification
  ○ 2.2 PaymentMethod
  ○ 2.3 BatchBooking
  ○ 2.4 NumberOfTransactions
  ○ 2.5 ControlSum
  ○ 2.6 PaymentTypeInformation
  ○ 2.18 RequestedCollectionDate
  ○ 2.19 Creditor
  ○ 2.20 CreditorAccount
  ○ 2.21 CreditorAgent
  ○ 2.22 CreditorAgentAccount
2.23 UltimateCreditor
2.24 ChargeBearer
2.25 ChargesAccount
2.26 ChargesAccountAgent
2.27 CreditorSchemeIdentification
2.28 DirectDebitTransactionInformation

Rule(s):

- ChargeBearerRule
  - If ChargeBearer is present, then DirectDebitTransactionInformation/ChargeBearer is not allowed.
  - If DirectDebitTransactionInformation/ChargeBearer is present, then ChargeBearer is not allowed.
  - DirectDebitTransactionInformation/ChargeBearer and ChargeBearer may both be absent.

- ChargesAccountAgentRule
  - If ChargesAccountAgent is present, then it must contain a branch of the CreditorAgent. It must not contain a completely different agent.

- ChargesAccountRule
  - If ChargesAccountAgent is present, then ChargesAccount must be present.

- CreditorSchemeIdentificationRule
  - If CreditorSchemeIdentification is present, then DirectDebitTransactionInformation/CreditorSchemeIdentification is not allowed.
  - If DirectDebitTransactionInformation/CreditorSchemeIdentification is present, then CreditorSchemeIdentification is not allowed.
  - DirectDebitTransactionInformation/CreditorSchemeIdentification and CreditorSchemeIdentification may both be absent.

- PaymentTypeInformationRule
  - If PaymentTypeInformation is present, then DirectDebitTransactionInformation/PaymentTypeInformation is not allowed.

- UltimateCreditorRule
  - If UltimateCreditor is present, then DirectDebitTransactionInformation/UltimateCreditor is not allowed.
  - If DirectDebitTransactionInformation/UltimateCreditor is present, then UltimateCreditor is not allowed.
  - DirectDebitTransactionInformation/UltimateCreditor and UltimateCreditor may both be absent.
Guideline(s):
  ○ UltimateCreditorGuideline
    ▪ UltimateCreditor may only be present if different from Creditor.

2.1 PaymentInformationIdentification <PmtInfId>
- Presence: [1..1]
- Definition: Unique identification, as assigned by a sending party, to unambiguously identify the payment information group within the message.
- Data Type: Max35Text
- Format: maxLength: 35, minLength: 1

2.2 PaymentMethod <PmtMtd>
- Presence: [1..1]
- Definition: Specifies the means of payment that will be used to move the amount of money.
- Data Type: Code
- One of the following PaymentMethod2Code values must be used:
  ○ DD DirectDebit: Collection of an amount of money from the debtor's bank account by the creditor. The amount of money and dates of collections may vary.

2.6 PaymentTypeInformation <PmtTpInf>
- Presence: [0..1]
- Definition: Set of elements used to further specify the type of transaction.
- Type: This message item is composed of the following PaymentTypeInformation20 element(s):
  ○ 2.7 InstructionPriority
  ○ 2.8 ServiceLevel
  ○ 2.11 LocalInstrument
  ○ 2.14 SequenceType
  ○ 2.15 CategoryPurpose

2.8 ServiceLevel <SvcLvl>
- Presence: [0..1]
- Definition: Agreement under which or rules under which the transaction should be processed.
- Type: This message item is composed of one of the following ServiceLevel8Choice element(s):
  ○ 2.9 Code
2.10 Proprietary

2.9 Code <Cd>
- Presence: [1..1]
- This message item is part of choice 2.8 ServiceLevel.
- Definition: Specifies a pre-agreed service or level of service between the parties, as published in an external service level code list.
- Data Type: ExternalServiceLevel1Code
- Format: maxLength: 4, minLength: 1

2.11 LocalInstrument <LclInstrm>
- Presence: [0..1]
- Definition: User community specific instrument.
- Usage: This element is used to specify a local instrument, local clearing option and/or further qualify the service or service level.
- Type: This message item is composed of one of the following LocalInstrument2Choice element(s):
  - 2.12 Code
  - 2.13 Proprietary

2.12 Code <Cd>
- Presence: [1..1]
- This message item is part of choice 2.11 LocalInstrument.
- Definition: Specifies the local instrument, as published in an external local instrument code list.
- Data Type: ExternalLocalInstrument1Code
- Format: maxLength: 35, minLength: 1

2.14 SequenceType <SeqTp>
- Presence: [0..1]
- Definition: Identifies the direct debit sequence, such as first, recurrent, final or one-off.
- Data Type: Code
- When this message item is present, one of the following SequenceType1Code values must be used:
  - FNAL Final Final collection of a series of direct debit instructions.
  - FRST First First collection of a series of direct debit instructions.
  - OOFF OneOff Direct debit instruction where the debtor's authorisation is used to initiate one single direct debit transaction.
○ RCUR Recurring Direct debit instruction where the debtor’s authorisation is used for regular direct debit transactions initiated by the creditor.

2.18 RequestedCollectionDate  <ReqdColltnDt>
● Presence: [1..1]
● Definition: Date and time at which the creditor requests that the amount of money is to be collected from the debtor.
● Data Type: ISODate

2.19 Creditor  <Cdtr>
● Presence: [1..1]
● Definition: Party to which an amount of money is due.
● Type: This message item is composed of the following PartyIdentification32 element(s):
  ○ 9.1.0 Name
  ○ 9.1.12 Identification

2.20 CreditorAccount  <CdtrAcct>
● Presence: [1..1]
● Definition: Unambiguous identification of the account of the creditor to which a credit entry will be posted as a result of the payment transaction.
● Type: This message item is composed of the following CashAccount16 element(s):
  ○ 1.1.0 Identification
  ○ 1.1.8 Type
  ○ 1.1.11 Currency
  ○ 1.1.12 Name

1.1.0 Identification  <Id>
● Presence: [1..1]
● Definition: Unique and unambiguous identification for the account between the account owner and the account servicer.
● Type: This message item is composed of one of the following AccountIdentification4Choice element(s):
  ○ 1.1.1 IBAN
  ○ 1.1.2 Other

1.1.1 IBAN  <IBAN>
● Presence: [1..1]
- This message item is part of choice 1.1.0 Identification.
- Definition: International Bank Account Number (IBAN) - identifier used internationally by financial institutions to uniquely identify the account of a customer. Further specifications of the format and content of the IBAN can be found in the standard ISO 13616 "Banking and related financial services - International Bank Account Number (IBAN)" version 1997-10-01, or later revisions.
- Data Type: IBAN2007Identifier
- Format: [A-Z]{2,2}[0-9]{2,2}[a-zA-Z0-9]{1,30}
- Rule(s):
  - IBAN
    - A valid IBAN consists of all three of the following components: Country Code, check digits and BBAN.

2.21 CreditorAgent <CdtrAgt>
- Presence: [1..1]
- Definition: Financial institution servicing an account for the creditor.
- Type: This message item is composed of the following BranchAndFinancialInstitutionIdentification4 element(s):
  - 6.1.0 FinancialInstitutionIdentification
  - 6.1.25 BranchIdentification

6.1.0 FinancialInstitutionIdentification <FinInstnId>
- Presence: [1..1]
- Definition: Unique and unambiguous identification of a financial institution, as assigned under an internationally recognised or proprietary identification scheme.
- Type: This message item is composed of the following FinancialInstitutionIdentification7 element(s):
  - 6.1.1 BIC
  - 6.1.2 ClearingSystemMemberIdentification
  - 6.1.7 Name
  - 6.1.8 PostalAddress
  - 6.1.19 Other

6.1.1 BIC <BIC>
- Presence: [0..1]
- Definition: Bank Identifier Code. Code allocated to financial institutions by the Registration Authority, under an international identification scheme, as described
in the latest version of the standard ISO 9362 Banking (Banking telecommunication messages, Bank Identifier Codes).

- **Data Type:** BICIdentifier
- **Format:** \([A-Z]\{6,6\}[A-Z2-9][A-NP-Z0-9]\{3,3\}\{0,1\}\
- **Rule(s):**
  - BIC
    - Valid BICs are registered with the ISO 9362 Registration Authority, and consist of eight (8) or eleven (11) contiguous characters comprising the first three or all four of the following components: BANK CODE, COUNTRY CODE, LOCATION CODE, BRANCH CODE. The bank code, country code and location code are mandatory, while the branch code is optional.

### 2.24 ChargeBearer <ChrgBr>
- **Presence:** [0..1], R1
- **Definition:** Specifies which party/parties will bear the charges associated with the processing of the payment transaction.
- **Data Type:** Code
- **When this message item is present, one of the following ChargeBearerType1Code values must be used:**
  - CRED BorneByCreditor All transaction charges are to be borne by the creditor.
  - DEBT BorneByDebtor All transaction charges are to be borne by the debtor.
  - SHAR Shared In a credit transfer context, means that transaction charges on the sender side are to be borne by the debtor, transaction charges on the receiver side are to be borne by the creditor. In a direct debit context, means that transaction charges on the sender side are to be borne by the creditor, transaction charges on the receiver side are to be borne by the debtor.
  - SLEV FollowingServiceLevel Charges are to be applied following the rules agreed in the service level and/or scheme.

### 2.28 DirectDebitTransactionInformation <DrctDbtTxInf>
- **Presence:** [1..n]
- **Definition:** Set of elements used to provide information on the individual transaction(s) included in the message.
- **Type:** This message item is composed of the following DirectDebitTransactionInformation9 element(s):
○ 2.29 PaymentIdentification
○ 2.32 PaymentTypeInformation
○ 2.44 InstructedAmount
○ 2.45 ChargeBearer
○ 2.46 DirectDebitTransaction
○ 2.69 UltimateCreditor
○ 2.70 DebtorAgent
○ 2.71 DebtorAgentAccount
○ 2.72 Debtor
○ 2.73 DebtorAccount
○ 2.74 UltimateDebtor
○ 2.75 InstructionForCreditorAgent
○ 2.76 Purpose
○ 2.79 RegulatoryReporting
○ 2.80 Tax
○ 2.81 RelatedRemittanceInformation
○ 2.88 RemittanceInformation

● Guideline(s):
  ○ UltimateCreditorGuideline
    ■ UltimateCreditor may only be present if different from Creditor.
  ○ UltimateDebtorGuideline
    ■ UltimateDebtor may only be present if different from Debtor.

2.29 PaymentIdentification  <PmtId>

- Presence: [1..1]
- Definition: Set of elements used to reference a payment instruction.
- Type: This message item is composed of the following PaymentIdentification1 element(s):
  - 2.30 InstructionIdentification
  - 2.31 EndToEndIdentification

2.31 EndToEndIdentification  <EndToEndId>

- Presence: [1..1]
- Definition: Unique identification assigned by the initiating party to unambiguously identify the transaction. This identification is passed on, unchanged, throughout the entire end-to-end chain.
- Usage: The end-to-end identification can be used for reconciliation or to link tasks relating to the transaction. It can be included in several messages related to the transaction.
- Data Type: Max35Text
- Format: maxLength: 35, minLength: 1

2.44 InstructedAmount  <InstdAmt>
- Presence: [1..1]
- Definition: Amount of money to be moved between the debtor and creditor, before deduction of charges, expressed in the currency as ordered by the initiating party.
- Data Type: ActiveOrHistoricCurrencyAndAmount
- This data type must be used with the following XML Attribute: Currency (Ccy) which is typed by ActiveOrHistoricCurrencyCode.
- Format: ActiveOrHistoricCurrencyAndAmount, fractionDigits: 5, minInclusive: 0, totalDigits: 18, ActiveOrHistoricCurrencyCode: [A-Z]{3,3}
- Rule(s):
  - CurrencyAmount
    - The number of fractional digits (or minor unit of currency) must comply with ISO 4217. Note: The decimal separator is a dot.
  - ActiveOrHistoricCurrency
    - The Currency Code must be registered, or have already been registered. Valid active or historic currency codes are registered with the ISO 4217 Maintenance Agency, consist of three (3) contiguous letters, and may be or not be withdrawn on the day the message containing the Currency is exchanged.

2.46 DirectDebitTransaction  <DrctDbtTx>
- Presence: [0..1]
- Definition: Set of elements providing information specific to the direct debit mandate.
- Type: This message item is composed of the following DirectDebitTransaction6 element(s):  
  - 2.47 MandateRelatedInformation
  - 2.66 CreditorSchemeIdentification
  - 2.67 PreNotificationIdentification
  - 2.68 PreNotificationDate

2.47 MandateRelatedInformation  <MndtRltdInf>
- Presence: [0..1]
- Definition: Set of elements used to provide further details of the direct debit mandate signed between the creditor and the debtor.
- Type: This message item is composed of the following MandateRelatedInformation6 element(s):
  - 2.48 MandateIdentification
  - 2.49 DateOfSignature
  - 2.50 AmendmentIndicator
  - 2.51 AmendmentInformationDetails
  - 2.62 ElectronicSignature
  - 2.63 FirstCollectionDate
  - 2.64 FinalCollectionDate
  - 2.65 Frequency
- Rule(s):
  - AmendmentIndicatorFalseRule
    - If AmendmentIndicator is false, then AmendmentInformationDetails is not allowed.
  - AmendmentIndicatorTrueRule
    - If AmendmentIndicator is true, then AmendmentInformationDetails must be present.

2.48 MandateIdentification  <MndtId>
- Presence: [0..1]
- Definition: Unique identification, as assigned by the creditor, to unambiguously identify the mandate.
- Data Type: Max35Text
- Format: maxLength: 35, minLength: 1

2.49 DateOfSignature  <DtOfSgntr>
- Presence: [0..1]
- Definition: Date on which the direct debit mandate has been signed by the debtor.
- Data Type: ISODate

2.50 AmendmentIndicator  <AmdmntInd>
- Presence: [0..1]
- Definition: Indicator notifying whether the underlying mandate is amended or not.
- Data Type: One of the following TrueFalseIndicator values must be used:
  - MeaningWhenTrue: True
  - MeaningWhenFalse: False

2.66 CreditorSchemeIdentification  <CdtrSchmId>
- **Presence:** [0..1], R4
- **Definition:** Credit party that signs the mandate.
- **Type:** This message item is composed of the following PartyIdentification32 element(s):
  - 9.1.0 Name
  - 9.1.1 PostalAddress
  - 9.1.12 Identification
  - 9.1.33 CountryOfResidence
  - 9.1.34 ContactDetails

**2.70 DebtorAgent <DbtrAgt>**
- **Presence:** [1..1]
- **Definition:** Financial institution servicing an account for the debtor.
- **Type:** This message item is composed of the following BranchAndFinancialInstitutionIdentification4 element(s):
  - 6.1.0 FinancialInstitutionIdentification
  - 6.1.25 BranchIdentification

**2.72 Debtor <Dbtr>**
- **Presence:** [1..1]
- **Definition:** Party that owes an amount of money to the (ultimate) creditor.
- **Type:** This message item is composed of the following PartyIdentification32 element(s):
  - 9.1.0 Name
  - 9.1.1 PostalAddress
  - 9.1.12 Identification
  - 9.1.33 CountryOfResidence
  - 9.1.34 ContactDetails

**2.73 DebtorAccount <DbtrAcct>**
- **Presence:** [1..1]
- **Definition:** Unambiguous identification of the account of the debtor to which a debit entry will be made as a result of the transaction.
- **Type:** This message item is composed of the following CashAccount16 element(s):
  - 1.1.0 Identification
  - 1.1.8 Type
  - 1.1.11 Currency
  - 1.1.12 Name
2.88 RemittanceInformation <RmtInf>
- Presence: [0..1]
- Definition: Information supplied to enable the matching of an entry with the items that the transfer is intended to settle, such as commercial invoices in an accounts' receivable system.
- Type: This message item is composed of the following RemittanceInformation5 element(s):
  - 2.89 Unstructured
  - 2.90 Structured

2.89 Unstructured <Ustrd>
- Presence: [0..n]
- Definition: Information supplied to enable the matching/reconciliation of an entry with the items that the payment is intended to settle, such as commercial invoices in an accounts' receivable system, in an unstructured form.
- Data Type: Max140Text
- Format: maxLength: 140, minLength: 1

9.1.0 Name <Nm>
- Presence: [0..1]
- Definition: Name by which a party is known and which is usually used to identify that party.
- Data Type: Max140Text
- Format: maxLength: 140, minLength: 1

9.1.12 Identification <Id>
- Presence: [0..1]
- Definition: Unique and unambiguous identification of a party.
- Type: This message item is composed of one of the following Party6Choice element(s):
  - 9.1.13 OrganisationIdentification
  - 9.1.21 PrivateIdentification

9.1.13 OrganisationIdentification <OrgId>
- Presence: [1..1]
- This message item is part of choice 9.1.12 Identification.
- Definition: Unique and unambiguous way to identify an organisation.
• Type: This message item is composed of the following OrganisationIdentification4 element(s):
  ○ 9.1.14 BICOrBEI
  ○ 9.1.15 Other

9.1.15 Other  <Othr>
• Presence: [0..n]
• Definition: Unique identification of an organisation, as assigned by an institution, using an identification scheme.
• Type: This message item is composed of the following GenericOrganisationIdentification1 element(s):
  ○ 9.1.16 Identification
  ○ 9.1.17 SchemeName
  ○ 9.1.20 Issuer

9.1.16 Identification  <Id>
• Presence: [1..1]
• Definition: Identification assigned by an institution.
• Data Type: Max35Text
• Format: maxLength: 35, minLength: 1

9.1.21 PrivateIdentification  <PrvtId>
• Presence: [1..1]
• This message item is part of choice 9.1.12 Identification.
• Definition: Unique and unambiguous identification of a person, eg, passport.
• Type: This message item is composed of the following PersonIdentification5 element(s):
  ○ 9.1.22 DateAndPlaceOfBirth
  ○ 9.1.27 Other

9.1.27 Other  <Othr>
• Presence: [0..n]
• Definition: Unique identification of a person, as assigned by an institution, using an identification scheme.
• Type: This message item is composed of the following GenericPersonIdentification1 element(s):
  ○ 9.1.28 Identification
  ○ 9.1.29 SchemeName
  ○ 9.1.32 Issuer
9.1.28 Identification  <Id>

- Presence: [1..1]
- Definition: Unique and unambiguous identification of a person.
- Data Type: Max35Text
- Format: maxLength: 35, minLength: 1

9.1.29 SchemeName  <SchmeNm>

- Presence: [0..1]
- Definition: Name of the identification scheme.
- Type: This message item is composed of one of the following PersonIdentificationSchemeName1Choice element(s):
  - 9.1.30 Code
  - 9.1.31 Proprietary

9.1.30 Code  <Cd>

- Presence: [1..1]
- This message item is part of choice 9.1.29 SchemeName.
- Definition: Name of the identification scheme, in a coded form as published in an external list.
- Data Type: ExternalPersonIdentification1Code
- Format: maxLength: 4, minLength: 1