Bridging the gap between Analog, Digital and Verifiable Credentials at an Higher Education Institution

1.1 Introduction

Verifiable Credentials (VCs) applied to the educational domain allow the student to receive one or more certificates of accomplishment (ranging from a micro credit up to a degree-granting diploma), to store them in the student's digital wallet (an app running in the smartphone), and to share a certificate with third-parties in such a way that the user has full-control on what gets shared and with whom. Each VC contains a set of attributes, it is possible to share only a subset of them, or to make calculations using them and only to share the result of that calculation.

Here is a simple example of the use and power of a VC. The student Alice, which looks much younger than she really is, requested and obtained her citizen card as a VC, and now wants to enter into a bar that has a minimum-age requirement. Alice opens the digital wallet app, selects the citizen card certificate, adds a checkmark to her `Picture` attribute, selects her `BirthDate` attribute, chooses *use a boolean function*, and by mostly pointing and clicking (or more likely, reusing a saved expression that she created earlier), creates the function "`add(BirthDate, years=18) <= Today()`", and then presses generate QR Code. The Digital Wallet app generates a QR Code, that she shows the bouncer Bob at the door. Bob uses a special App on his smartphone to read the QR Code, and obtains back two attributes: her `Picture` and the information that there is a guarantee[1] she is over-18. Bob lets her in, without coming to see her identity card nor being able to see and memorize her full name and birth-date. As an added benefit Bob can archive the VC that he has just obtained, to later prove that he didn't let her in without first checking that she was elegible for entry.

The main idea behind the use of verifiable credentials (put the user in control of the information and of the decision of what to share) is simple to understand, in privacy terms the power can be huge, but the technology is complex, not yet stable, and not easy to integrate with existing information systems and to deploy into production.

This document describes and implements an open-source digital platform, appropriate for self-hosting or deploying in the cloud, that has three subcomponents — issue certificates, access certificates, interface with VCs technical stack —, that together can help support the transition from normal paper-based certificates to VCs, maintaining simultaneous support for both systems.

1.2 The Problem of supporting Fancy Certificates

The technical complexity of deploying into production and operating a reliable verifiable credentials infrastructure means that it is likely that most Higher Education Institutions (HEIs) will take a wait and see approach, and will delay participating into those kinds of initiatives, outside of isolated and limited scope pilot projects. Even if a pilot is successful it does not seem likely that in the near future the HEI will commit to production use.

Another problem is that there is a need to simultaneously support students (and third-parties) that can only use or prefer to receive a paper (or PDF) certificate, and people that see the advantages of receiving a Verifiable Credential (VC). In spite of the recent advances, powered by the pandemic that pushed a renewed interest in digital identity [Preukschat and Reed 2021], the technology is still in a new and immature state, and for the HEIs integrating it into the existing workflows and information systems seems to be a dangerous and expensive endeavour with a very bad ROI: most students, unless they are significantly privacy-conscious, will probably not care. Thus the risks seem to be too high and the benefits too low if the HEI discounts the bragging rights of being able to claim to be on the forefront of digital identity.

Thus there are at least these problems: complexity, cost, not-enough demand, technology uncertainty, analog vs digital.

[1] The way the guarantee works is similar to the way a Public-Key Infrastructure (PKI) works. The difference is that it is a descentralized infrastructure and, in our case, the European Blockchain Service Infrastructure (EBSI) is involved to provide legal guarantees at the european level.

1.3 A Low-Tech Solution

The low-tech approach proposed and implemented here to minimize some of the problems mentioned in the previous section, and to allow aiming for the "*the full monty*" right now, that is rushing into production use without waiting for the technology to stabilize, is no more than the combination of these eleven very simple ideas/options/insights:

1. Use PDF as the certificate delivery agent
2. Reduce information systems integration to the bare minimum
3. Leverage existing citizen-card based digital signatures
4. Provide value even for those that don't know or want VCs
5. Automate as much as possible
6. Embed the VCs inside the PDF
7. Guarantee long-term access to the certificate
8. Support reissuing and revoking certificates from day one
9. Try to be a catalyst for the network effect to take place
10. Support certificate delivery by email, or printing and sending by post or hand delivering
11. Support certificate recovery and going from digital to analog and back to digital

1.3.1 Use PDF as the certificate delivery agent

The PDF format is very useful for our purposes because:

- It is a standard [ISO 32000-1:2008(E) 2008], everyone is accustomed to using it, there are open source programs to process (parse) its contents or to create a PDF from scratch
- Supports embedding inside (attach) one or more files of different types, namely JSON, thus allowing the certificate contents to be much richer than what is seen in the printed version and to be suitable for automatic processing
- If we use the PDF/A variant suitable for long-term preservation of electronic documents (amongst other things it embeds all fonts inside), it is expected that the PDF will be readable a very long time in the future. Although embedding JSON files in a PDF/A file was initially forbidden it is allowed in PDF/A-3 [Arms et al. 2014; ISO 19005-3:2012(E) 2012]
- A PDF is usually the digital source from which certificates are printed. Even if, for the sake of argument, we could only get our hands on the final printed version, provided that we could *convince* the existing system used to generate the certificate to, before printing, add an image with a QR Code, we could at least in principle digitalize the resulting paper certificate, save it as PDF and attach the JSON files

1.3.2 Reduce information systems integration to the bare minimum

The largest difficulty of implementing support for issuing VCs is most likely to be integrating with the information system used by the HEI. Reducing the level of integration reduces cost and simplifies things, later on it should be possible, once the problem is well understood, to move in the opposite direction and better integrate with that information system.

The first 238 micro course credentials issued by UMAIA (notice that they are PDF's with embedded JSON files that are not VCs, VCs support will come later) used this simple integration technique:

- It was requested, just for the relevant students, an export in Excel format from the university information system of this information: date of birth, country of origin. For a subset of those students it was also obtained the student private email (only for those students that it was likely that they did not read their school email)

- It was requested for the micro courses coordinator to be added to the list of teachers of the 5 micro courses so that he could directly download the corresponding student grades in Excel format from the university portal

- The attendance rate of each student in each micro course edition is not stored in the UMAIA's information system, thus that value was obtained by analyzing the attendance sheets that the students signed, and calculating its value

- The 5 micro courses "*program sheets*" (that contain the learning outcomes, syllabus, etc.) were downloaded in PDF format (one in EN and another in PT) from the university portal and using copy and paste and some minor editing added to the Django model MicroCourse instances using the Django admin interface

- The information concerning the 5 micro courses; 18 micro course editions; 329 students name, date of birth and country of origin, and for each course and each student, the student's attendance rate (that is, everything except the information extracted from the courses "*program sheets*" and the information about the student's grades), was organized into a single Excel file with 4 Excel sheets (Courses, Editions, Students, Attendance) that were linked using the Excel `vlookup()` function. That file was then imported into the application using the Django admin interface. The fields are (underlined fields use Excel formulas, namely `vlookup()` and text concatenation):

  1. `Courses: course_code, title_pt`
  2. `Editions: CourseEditionCode, course_code,` <u>`title_pt`</u>`, assessment_type, start_date, end_date, stackability, instruction_language, school_class, course_coordinator`
  3. `Students: student_number, student_name, given_name, family_name, date_of_birth, sex, country, student_email, private_email`
  4. `Attendance: CourseEditionCode,` <u>`CourseEditionTxt`</u>`, student_number,` <u>`student_name`</u>`, funding_program, attendance_percentage`

- It was requested for the domain "`certificados.umaia.pt`" to be registered at the university DNS and that it pointed to the public IP address of the virtual machine where the software was installed

The integration improvements that we intend to implement before trying to use this system for issuing certificates for all the university courses are:

- To be able to obtain the courses "*program sheets*" in JSON format, together with a separate file with an information system issued signature that validates the JSON contents

- To be able to obtain the student grades in JSON format, together with a separate file with an information system issued signature that validates the JSON contents

For the Event Certificate use case (see Section 1.4.1) the information can be input directly in the Django admin and any structured information that might exist (e.g. obtained from the event registration form) can be imported using an Excel file.

1.3.3 Leverage existing citizen-card based digital signatures

Paper based certificates commonly use heavier grammage paper and the HEI's *white stamp / white seal* that instead of using ink creates an embossed mark on the paper. One problem of this format is that in a digitezed version of that document the stamp is usually invisible[2]. Even if ink stamps are used and the ink is visible it is always a problem to use a digitized version of a certificate because nowadays everyone has access to tools for editing the resulting PDF, thus the document is very easy to modify which facilitates fraud. Two solutions to this problem are commonly used: qualified digital signatures or printing in the certificate a code that can be used online in the HEI's web site to validate the certificate. Here we are going to use these two solutions simultaneously.

For now the student grades exported from the HEI's information system are not yet accompanied by a detached digital signature generated by the HEI's information system that validates the data and demonstrates that the grades were not tampered with, that is expected to be added later. In our case the grades are exported as an Excel file that identifies the curricular unit and the school class, and that for each student in that class specifies the student number, name and grade. The digital platform parses that Excel file to create a 2-or-more pages PDF that: 1) presents the grades in a table, 2) has embedded inside the PDF the original Excel file and a JSON rendering of that Excel file, 3) specifies when that Excel file was downloaded from the Information System[3], 4) adds the name and function or authority of who is going to sign (for the time being we are using the course coordinator, later on it will be someone from the academic support staff) with a visible qualified digital signature[4]. In this way the person that downloaded the grades from the HEI's Information System takes full responsibility that the information was not changed after the file was downloaded.

It will be important in the future for the HEI Information System to add a detached digital signature to the exported grades file because it might happen that sometime in the future the information might differ, and we need to assign responsibility and to find out what happened:

- In a normal situation (which will probably account for more than 99.9% of the cases) the student gets a grade in a curricular unit, that grade is exported from the HEI's Information System and used in the certificate, and one or two years later the grade will appear in the "*livro de termos*", a bound book that aggregates all grades from all the students and that works as the definitive paper support for each grade.

- The problem is that if a mistake is later found out after the certificate was issued and delivered to the student, and the grade needs to be corrected, then (in our case) there is a procedure for the teacher to request that the grade be modified, and if that request is approved the grade will be corrected. Now, if that happens, the grade present in the HEI's Information System and in the "*livro de termos*" is different from the grade present in the already issued certificate.

The advantage of adding a detached digital signature generated by the HEI's Information System is that it shows that although in that specific case the data in the certificate is different from the data in the HEI's Information System and "*livro de termos*", it was not the person that signed the PDF with the grades that changed that specific grade.

The system for each certificate uses a 2-level chain of responsibility:

- The course coordinator exports the grades and signs the PDF with a digital qualified signature, thus taking responsibility that the grades were not changed after exporting

- The Rector signs the certificates with a digital qualified signature using as support the grades signed by the course coordinator. This is efficient because the desktop version (Windows, macOS, Linux) of the Portuguese government `autenticação.gov.app` supports bulk signing of certificates provided the visible signature is to appear in the same place, that is why the signature is always added to a fixed spot in the first page of the PDF.

[2] `https://www.provedor-jus.pt/en/authentication-of-documents-ombudsmans-office-suggests-that-parish-councils-offer-citizens-an-alternative-to-using-the-white-stamp/`

[3] That information is included inside the Excel file, if it wasn't it would not be possible to obtain.

[4] Obtained by e.g. using a mobile phone together with the Portuguese Digital Mobile Key, or by using the Portuguese citizen card together with a smart card reader.

1.3.4 Provide value even for those that don't know or want VCs

To best way to foster adoption and visibility of the format proposed in this document is to make sure that even those that cannot or do not want to use VCs will be interested in the contents of the new certificates. The idea is to embed inside the PDF additional information in JSON format, namely:

- For a certificate of attendance to an event: detailed information about the event, see Section 1.4.1
- For a micro course certificate: detailed information in PT and EN about the course, including its syllabus, see Section 1.4.2
- For a multi-year course certificate: detailed information in PT and EN about all the curricular units of the course, see Section 1.4.5

In Portugal it would be PT and EN, in another country it would be the national language and EN (to simplify the system only a national language is supported, if EN was the national language the second language could be disabled).

The information embedded inside the PDF can be visualized by the user ou automatically extracted and processed by a program. The idea is to use the following 3 or 4 embedded files with equivalent contents:

- A text file in markdown[5] format that can be visualized by an human being and printed (this file did not exist in the first 238 certificates issued by UMAIA in April 2025)
- An ad-hoc JSON file that can be automatically processed
- An ELM v3 JSON-LD file that can be automatically processed and that complies with standard schemas (this file did not exist in the first 238 certificates issued in April 2025)
- A VC JSON-LD file, if the user already finished the onboarding process and thus can use verifiable credentials (this file did not exist in the first 238 certificates issued in April 2025)

One of the interesting use cases is a student that studied in a given university, did not graduate, and a few years later wants to continue studies in a new university. The new university can extract from the certificate detailed information about the curricular units that the student actually attended and got a passing grade (even if the course was in the mean time altered or stopped being offered and thus consulting the university web site provides different information), making it much easier to give the appropriate credits and grades for the equivalent courses that the student will not be required to attend. There is no need to contact the old university to obtain that information.

1.3.5 Automate as much as possible

To simplify as much as possible the process of issuing certificates it should be possible:

1. To choose a course, or to choose a course and a time range, or to choose a student, and to issue all the corresponding certificates that hadn't yet been issued
2. To download a ZIP file with the set of certificates that need to be signed. These certificates can be bulk signed in a single or a couple of operations by using the desktop version (Windows, macOS, Linux) of the Portuguese government `autenticação.gov.app` that supports bulk signing of certificates, provided the visible signature is to appear in all files in the same place.

[5] `https://www.markdownguide.org/basic-syntax/`

3. To upload a ZIP file with a set of signed certificates, the certificates must be automatically extracted from the ZIP file and placed at the appropriate locations (defined by the URL associated with its QR Code). At this moment a mail merge Excel file is also generated to help the academic staff send an email to the students providing the URL from where to download the certificate. That mail merge should support sending to the private email address of the student (if it was provided to the system) and to the normal HEI email address of the student

The April 2025 version of the software used to issue 238 micro course certificates implemented these three features. The only limitation of that version was that each operation had to be done for a given school class, it was not possible to group more than one school class in the same operation, thus 18 operations had to be made. This limitation is expected to be removed.

1.3.6 Embed the VCs inside the PDF

By embedding the VCs inside the PDF certificate we are making sure all the certificates, those with VCs and those without VCs, are delivered and maintained in exactly the same way. It is up to the student to extract the VC from inside the PDF of the certificate and add it to the student's digital wallet.

Our goal is to use the Django app (not yet implemented) that interacts with the DC4EU technical stack to issue and obtain the VCs for those students that finished the onboarding process.

Even if the certificate was already issued without a VC it is possible to issue it again (marking it as `v2` and this time to embed a VC credential within the PDF). It will be also possible, using the support for superseding or revoking a certificate, to later on, as the technology evolves, to improve the compatibility of the VCs or to replace or migrate the student VCs to VCs in a different format.

1.3.7 Guarantee long-term access to the certificate

Long term access is implemented by using permanent URLs. Besides storing the certificates in fixed locations[6] of the file system of the web server and using e.g. NGINX to provide access to them, we also need some form of warning the user that a certificate was superseded or revoked and to allow the users to obtain the certificate by just providing the certificate ID[7].

To support this we just need a simple key-value mapping application, that maps the certificate ID into the URL of the PDF. By being very simple it is easier to protect (it is the only part of the application that must be available in the internet, the other two components should be accessible only from the intranet or require the use of a VPN). It is also easier to replace sometime in the future if the requirements or the technology changes.

The first batch of 238 certificates issued in April 2025 used the following convention for the permanent URL (that now can no longer be changed), trying to use an URL that was not too long (in this case, depending on the length of the student first and last names, it was a little over 100 chars): `https://`*certificatesDomainName*`/media/certificates/`*yyyy/mm/dd/hhmmss*`/cert-`*StudentNumber-StudentFirstLastName-9charUuidSuffix*`-s.pdf`, for instance `https://certificados.umaia.pt/media/certificates/2025/04/05/182342/cert-48249-MartaSilva-75ea7b988-s.pdf` (this is not a link to an existing certificate thus do not try to use it)

certificatesDomainName: in this case it used `certificados.umaia.pt`, any appropriate HEI domain name could be used

media/certificates: to facilitate making a backup all files generated by using the software system are always placed under the `media` directory, to separate them from other files generated by the system that might not be certificate specific, we used the `certificates` subdirectory

[6]Although from the outside the URLs are fixed, from the inside they are easy to change and adapt: the server name is controleed by a DNS entry, the pathnames can be adjusted by using e.g. symbolic links.

[7]There are three equivalent ways to obtain the certificate: by using the QR Code, by using the full permanent URL of the certificate, or by using the certificate ID (32 hexadecimal-digit UUID) together with the name of the web server (in our case `certificados.umaia.pt`)

yyyy/mm/dd/hhmmss: the date and time the certificate was issued. Grouping them by date might help in searching for a certificate, doing incremental backups, and avoids that the certificates be all of them in the same directory which in the long run could cause some performance impact

StudentNumber: the student number used at the HEI. The JSON embedded inside the PDF uses the European Student Identifier, that in this case was created by prefixing the student number with the fixed string prefix "`urn:schac:personalUniqueCode:int:esi:maieutica.pt:`", as defined in `https://wiki.geant.org/display/SM/European+Student+Identifier`

StudentFirstLastName: the student first and last name, both words with the first letter capitalized and the remaining in lowercase[8]

9charUuidSuffix: the certificate ID is a random 32 hexadecimal-digit UUID, here in the certificate filename we only use the last 9 hexadecimal digits of the certificate UUID

-s: we use the `-s` suffix for all files that have a qualified digital signature, in this case the Rector signed those files

To guess the certificate URL and thus obtain the certificate (which should be considered a privacy violation) it is necessary to know the student number, the student first and last name, the approximate date and time when the certificate was generated (possible if you happened to belong to the same school class and you also got a certificate), and to brute force guess the 9 hexadecimal digits: $16^9 = 2^{36}$ possibilities, 68 719 476 736 attempts, or half of them to have a 50% chance of success. This can be stopped by e.g. temporarily blocking requests that come from IP addresses that originated more than lets say one hundred failures in the same day (HTTP error 404: file not found) for a file with the `https://certificados.umaia.pt/media/certificates` prefix. This number must be low compared to the total number of possibilities because of the risk of the attacker using a botnet with multiple different IP addresses.

This filename convention makes it easy to manually or programmatically browse the certificate file system and extract the certificate that we are interested into and to reduce the likelihood that we might provide someone with the wrong certificate (which could happen if the mail merge support that is already implemented was not used, if it was used instead a manual operation, and if the certificate filename did not visually identify the student).

1.3.8 Support reissuing and revoking certificates from day one

To support superseding and revoking certificates the following solution is proposed:

1. A certificate ID is a 32 hexadecimal-digit UUID, using it in the web server interface always provides access to the latest version of the certificate.

2. The first version of a certificate has the string `v1` printed at the top right of the second page of the certificate. If it is necessary to reissue that certificate then the second version of the same certificate has the string `v2`, and so on if further versions are needed. This simplifies looking at the printed instances of two different versions of the same certificate and figuring out which one is the newer. Notice that the certificate has not an issuance date, the issuance date is the date of the digital signature (visible in the first page of the certificate).

3. The access URL, as obtained from the certificate QR code or from the email sent to the student, must be permanent, that is, it must still allow access to that version of the certificate, even if in the mean time that certificate was reissued or revoked. As such, we opt for the following solution, whenever a certificate is reissued (it can easily happen because we want to update the embedded files) or revoked (it will seldom happen):

[8] This was not fully respected in the first batch of certificates issued. If the student name in the information systems database was all in caps, as some names were, then the *StudentFirstLast* was issued without changing the remaining letters to lowercase as it should have been done.

(a) The PDF that was at that address is moved to another location (e.g., for the example from the previous section, moved from `https://certificados.umaia.pt/media/certificates/2025/04/05/182342/cert-48249-MartaSilva-75ea7b988-s.pdf` to `https://certificados.umaia.pt/media/obsolete/2025/04/05/182342/cert-48249-MartaSilva-75ea7b988-s.pdf`). That location can and should be easily guessed by knowing the URL of the certificate that was superseded (reissued or revoked)

(b) At the URL of the old version of the certificate we place a new digitally signed[9] PDF, it says that the certificate was reissued or revoked and specifies what was changed between that and the newest version, providing the new URL to that old version of the certificate and the URL to the latest version of that certificate. To support automation the same information is embeded inside in a JSON file.

(c) A certificate issued n times requires that we store online for ever $2 \times n - 1$ signed PDFs. This can increase significantly the storage space requirements if we keep on reissuing certificates, still it does not seem it will become an issue.

(d) In an extreme case, if for example certificate `v1` was superseded by `v2` (e.g. fix student birth date due to a database error), and then superseded by `v3` (e.g. embed a VC after student onboarding), and superseded again by `v4` (e.g. update the VC for technical reasons), then we have the following:

   i. If we use the certificate ID we only get access to the PDF of the latest version of the certificate (`v4`). If the certificate was revoked that is specified in the PDF, otherwise the PDF does not mention anything about the existence of `v1`, `v2`, `v3`, although, as it has a `v4` visibly printed, it is obvious it is the fourth version issued

   ii. If we use the URL for `v1` then we have access to the PDF that specifies what was changed from `v1` to `v4`, and allows access to `v1` and to `v4`

   iii. If we use the URL for `v2` then we have access to the PDF that specifies what was changed from `v2` to `v4`, and allows access to `v2` and to `v4`

   iv. If we use the URL for `v3` then we have access to the PDF that specifies what was changed from `v3` to `v4`, and allows access to `v3` and to `v4`

   v. If we use the URL for `v4` then we have access to the PDF of the certificate

The reasons why, when a certificate is superseded, we still allow for those that knew the old URL of a given version of a certificate access to the corresponding old version of that certificate are:

- Guarantee certificate recovery and the ability to access the same version for verification purposes or for legal reasons (e.g. to prove that at that time that certificate was valid). Although we do not provide the dates that correspond to the intermediate versions of the certificate all the changes are listed.

- If someone made a software application that was able to parse and integrate data from one of the embeded files present in the old version of the certificate, then we provide the updated file but we still provide access to the old file so that it can still be processed

From what was presented and discussed above, we can conclude that there is no reason for someone that had access to `v2` to get access to `v3` if it never tried to access `v2` at a time when `v4` did not yet exist. This also means that when `v4` is issued we have to update the PDFs that are at the URLs of `v1`, `v2` and `v3`. One corner case remains to be discussed: what needs to be done if a certificate is revoked and then e.g. after a legal battle is reissued again? Nothing, we do exactly as above.

Warning: by placing a different PDF at the exact same address the browser cache or the proxy cache might return the old version of the certificate even if that PDF no longer exists at the server. The solution is to set a relatively small CDN TTL (Contents Delivery Network Time to Live), e.g. 24 hours.

[9] It is signed by the same person or a person with the same role as the one that is signing the certificates.

1.3.9 Try to be a catalyst for the network effect to take place

The utility of the system proposed here is directly affected by the quantity of HEIs using it, namely its use for reducing the administrative overhead and helping giving credit for a partial course that a student took in another HEI only works if that other HEI is also using a similar system, ideally including an embedded JSON-LD file in ELM v3 format.

To facilitate adoption, the software is to be provided as an open source simple to use and well documented reusable Django app, that includes 3 subapps, that can be deployed together or separately, and that provide support for issuing credentials, accessing credentials, and interfacing with a VC credential issuance system (in our case the DC4EU support for issuing VCs supported by EBSI).

1.3.10 Support certificate delivery by email, or printing and sending by post or hand delivering

If the person for whom the certificate was issued does not have or did not provide an email address then the certificate can be printed and, if a home or professional address was provided, sent by post, otherwise it can be hand delivered to the user.

The support for mail merge allows sending an email that does not include the certificate itself, rather it provides a download URL for the certificate. If even that is problematic (e.g. embedded links are removed for security reasons or cause the email to be blocked) then it is possible to just provide the certificate UUID and say where it can be obtained using a base URL that is just the webserver name (without any links).

1.3.11 Support certificate recovery and going from digital to analog and back to digital

The following is supported:

- Obtaining the PDF of the certificate, printing the certificate, and losing aceess to the PDF file. If that happened then, using the QR code or the certificate ID, the PDF can be recovered (for 2-page certificates having access to the first or to the second page is enough to recover the PDF of the certificate).
- The student or event participant does not have an email address or digital device: the certificate is printed by the HEI and sent by post, or hand delivered to the user. If that happened then, using the QR code or the certificate ID, a couple of days, months or years later the PDF can be obtained
- Knowing just the certificate ID or knowing the URL of the certificate: from there it is possible to obtain the PDF

All the above examples are for self-service access, no interaction with the HEI is necessary. If on the other hand a printed version is not available, and neither the URL nor the certificate ID is known, then it is necessary to interact with the HEI in order to obtain access to the certificate[10].

1.4 Types of Certificates

The goal is to at least support these five types of certificates:

1. Event certificate
2. Micro Course certificate

[10] If the student already onboarded for the use of VCs then it might be possible to solve this using self service without the need for interacting with the HEI.

3. Learning Path certificate (two or more micro courses)
4. Certificate for curricular units of an incomplete multi-year Course
5. Certificate for a completed multi-year degree granting Course

In Portugal for the case of primary and secondary education the format of the certificates and diplomas is defined in [Portaria194/ 2021].

1.4.1 Certificate issued for attending an Event

The certificate is a file in PDF format that contains visible text and vector images and a set of text[11] files embedded as attachments:

Visible: 1-page A4 certificate in portrait layout with credential UUID, QR Code for downloading the PDF certificate, person name, event name, location, date, issuing entity logotype image, issuing entity name, name and function or authority of who made that signature, visible qualified digital signature[12]

Embedded: everything present in the PDF visible part plus the event description and program in PT and EN in three or four UTF-8 encoded attached text files with equivalent text contents: 1) printable text in markdown format with line breaks, 2) JSON, 3) JSON-LD, 4) VC (optional, only if the user already did the onboarding process)

1.4.2 Certificate issued for short duration course (Micro Course)

The certificate is a file in PDF format that contains visible text and vector images and a set of text files embedded as attachments:

Visible: 2-page A4 certificate in portrait layout:

First page of PDF ⇒ issuing entity name and logo, course name, person name, QR Code for downloading the PDF certificate, name and function or authority of who made that signature, visible qualified digital signature

Second page of PDF ⇒ **issuing entity** (name, address, logo); **course** (title, description, learning outcomes, ECTS, start and end date, European Qualifications Framework Level, language of learning and assessment, type of quality assurance, type of assessment, grade); **person** (name, european student identifier, date of birth, country); **credential** (UUID, number of times it was issued[13], URL for further information and downloading the certificate)

Embedded: course description, learning outcomes, and syllabus in PT and EN in three or four UTF-8 encoded attached text files with equivalent text contents: 1) printable text in markdown format with line breaks, 2) JSON, 3) JSON-LD, 4) VC (optional, only if the user already did the onboarding process)

[11] Even when it is a JSON file a `.txt` extension is used as it facilitates opening the file in a *normal* computer

[12] Obtained by e.g. using a mobile phone together with the Portuguese Digital Mobile Key, or by using the Portuguese citizen card together with a smart card reader. The desktop version (Windows, macOS, Linux) of the Portuguese government `autenticação.gov.app` supports bulk signing of certificates, provided the visible signature is to appear in all files in the same place.

[13] The value `v1` for the first time, `v2` means it was issued again with some corrections (e.g. correcting the spelling of the person name or some other information that later on was identified as incorrect). When confronted with two printed versions of the same certificate (same certificate UUID) that are not identical, the one with the larger `v#` where `#` is a positive integer, supersedes the other.

1.4.3 Certificate issued for Learning Path (two or more Micro Courses)

Visible: 2 or more pages A4 certificate in portrait layout:

First page ⇒ issuing entity name and logo, learning path name, person name, QR Code for downloading the PDF certificate, name and function or authority of who made that signature, visible qualified digital signature

Second and following pages ⇒ **issuing entity** (name, address, logo); **learning path** (name, total number of ECTS, ECTS-weighted average grade, start date of first course and end date of last course), list of **courses** (title, microcredential certificate UUID, grade); **person** (name, european student identifier, date of birth, country); **credential** (UUID, number of times it was issued, URL for further information and downloading the certificate)

Embedded: for each course, the course description, learning outcomes, and syllabus in PT and EN in three or four UTF-8 encoded attached text files with equivalent text contents: 1) printable text in markdown format with line breaks, 2) JSON, 3) JSON-LD, 4) VC (optional, only if the user already did the onboarding process)

1.4.4 Certificate issued for Curricular Units of an incomplete Course

Visible: 2 or more pages A4 certificate in portrait layout:

First page ⇒ issuing entity name and logo, course name, branch ou course specialization name if applicable, information that the course is only partially completed, person name, QR Code for downloading the PDF certificate, name and function or authority of who made that signature, visible qualified digital signature

Second and following pages ⇒ **issuing entity** (name, address, logo); **course** (name, total number of ECTS, total number of ECTS already obtained, ECTS-weighted average grade, start date of first course and end date of last course), list of **curricular units** (title, year/semester, school year of the grade, grade); **person** (name, european student identifier, date of birth, country); **credential** (UUID, number of times it was issued, URL for further information and downloading the certificate)

Embedded: for each course, the course description, learning outcomes, and syllabus in PT and EN in three or four UTF-8 encoded attached text files with equivalent text contents: 1) printable text in markdown format with line breaks, 2) JSON, 3) JSON-LD, 4) VC (optional, only if the user already did the onboarding process)

1.4.5 Certificate issued for a completed multi-year Course

Visible: 2 or more pages A4 certificate in portrait layout:

First page ⇒ issuing entity name and logo, course name, branch ou course specialization name if applicable, information that the course was completed (and in the case of a degree granting course the name of the degree), person name, QR Code for downloading the PDF certificate, name and function or authority of who made that signature, visible qualified digital signature

Second and following pages ⇒ **issuing entity** (name, address, logo); **course** (name, total number of ECTS, ECTS-weighted average grade, start date of first course and end date of last course), list of **curricular units** (title, year/semester, school year of the grade, grade); **person** (name, european student identifier, date of birth, country); **credential** (UUID, number of times it was issued, URL for further information and downloading the certificate)

Embedded: the diploma supplement in PT and EN and, for each course, the course description, learning outcomes, and syllabus in PT and EN, in three or four UTF-8 encoded attached text files with equivalent text contents: 1) printable text in markdown format with line breaks, 2) JSON, 3) JSON-LD, 4) VC (optional, only if the user already did the onboarding process)

This PDF Certificate includes all the relevant information for the complete, diploma or degree granting course (multi-year, several curricular units, branches, optional curricular units, diploma supplement).

1.5 JSON, JSON-LD and ELM v3

The goal is to fully support JSON-LD[14] and ELM[15] v3, the current version of the software only supports the ad hoc JSON presented in the next section.

1.5.1 Ad-hoc JSON

The 238 micro credential certificates issued in April 2025 used this JSON schema. This complied as much as possible with [Council of the European Union 2022].

14 ⟨*sample ad hoc JSOM scheme used for a microcredential of a standalone micro course* 14⟩≡

```
{
  "IssuingEntity": {
    "name": {
      "PT": "...",
      "EN": "..."
    },
    "address": "...",
    "zip_code": "...",
    "city": "...",
    "country": "...",
    "accrediting_body": {
      "PT": "...",
      "EN": "..."
    },
    "accreditation_start_date": "...",
    "accreditation_period": ...,
    "site": "...",
    "email": "..."
  },
  "MicroCourseEdition": {
    "type": "...",
    "course_code": "...",
    "title": {
      "PT": "...",
      "EN": "..."
    },
    "isced_code": "...",
    "description": {
      "PT": "...",
      "EN": "..."
    },
    "syllabus": {
      "PT": "...",
      "EN": "..."
```

[14] https://json-ld.org, https://www.w3.org/TR/json-ld/
[15] https://github.com/european-commission-empl/European-Learning-Model

```
    },
    "learning_outcomes": {
      "PT": "...",
      "EN": "..."
    },
    "ects": ...,
    "assessment": {
      "PT": "...",
      "EN": "..."
    },
    "assessment_format": {
      "PT": "...",
      "EN": "..."
    },
    "quality_assurance": {
      "PT": "...",
      "EN": "..."
    },
    "grading_scheme": {
      "PT": "...",
      "EN": "..."
    },
    "prerequisites": {
      "PT": "...",
      "EN": "..."
    },
    "eqf_level": 6,
    "assessment_type": {
      "PT": "...",
      "EN": "..."
    },
    "start_date": "...",
    "end_date": "...",
    "stackability": ...,
    "instruction_language": "...",
    "school_class": "...",
    "n_enrolled": ...,
    "n_finished": ...
  },
  "MicroLearner": {
    "type": "...",
    "student_number": ...,
    "European_Student_Identifier": "...",
    "given_name": "...",
    "family_name": "...",
    "date_of_birth": "...",
    "sex": "...",
    "country": "...",
    "student_email": "...",
    "is_steam": ...
  },
  "funding_program": "...",
  "grade": "...",
  "attendance_percentage": "...",
  "credential_id": "...",
  "credential_url": "...",
  "supersedes_credential_id": ...,
  "nw_version": "...",
```

```
    "schema_version": "1.00"
  }
```

Uses `IssuingEntity` 25.

We are using the `schema_version` to identify changes to this ad hoc JSON schema.

1.5.2 JSON-LD and ELM v3

Sample ELM v3 in JSON (unsigned) and JSON-LD (signed) formats are diffficult to find. Still, in `https://code.europa.eu/qualifications-courses-and-credentials/ELM-support/` (the files were uploaded in February 2025) there is an example for the fictional italian student Ana Andromeda studying in a Swedish university, with a certificate of participation (base64 encoded images[16]), a micro credential (also with base64 encoded images), a diploma for "Master of Arts in Media Studies" (the unsigned version of the sample file has 7874 lines and 571 KB, the signed version has 1 line and 473 KB), a joint[17] degree (sample mock credential joint degree for a master's of arts in media studies), and a transcript of records of a "Master of Arts in Media Studies" (14 626 lines and 2MB for the unsigned version, and 1 line and 1.3MB for the signed version).

Here is the ELM v3 mapping between some of the fields used in the ad hoc JSON file embedded inside the 238 micro credential PDF certificates issued by UMAIA in April 2025, including some fields that were not included but seem relevant. This mapping was created by browsing the European Learning Model Ontology[18], not by using the ELM sample files referenced above:

1. HEI
   (a) name: `http://data.europa.eu/snb/model/elm/Organisation`
   (b) address: `http://data.europa.eu/snb/model/elm/fullAddress`
   (c) country code: `http://data.europa.eu/snb/model/elm/countryCode`
   (d) email: `http://data.europa.eu/snb/model/elm/emailAddress`
   (e) web resource: `http://data.europa.eu/snb/model/elm/WebResource`
   (f) accreditation: `http://data.europa.eu/snb/model/elm/Accreditation`
   (g) accreditting body: `http://data.europa.eu/snb/model/elm/accreditingAgent`
   (h) accreditation jurisdiction: `http://data.europa.eu/snb/model/elm/limitJurisdiction`
   (i) accredited for EQF level: `http://data.europa.eu/snb/model/elm/limitEQFLevel`
   (j) report: `http://data.europa.eu/snb/model/elm/report`
2. Student
   (a) date of birth: `http://data.europa.eu/snb/model/elm/dateOfBirth`
   (b) family name: `http://xmlns.com/foaf/spec/#term_familyName`

[16] The certificate of participation is a set of PNG and JPEG images encoded in base64, to see the images copy the base64 string that is inside the quotation marks to e.g. `https://onlinepngtools.com/convert-base64-to-png` (in spite of its name it works for both PNG and JPEG files). In this case one of the images, a 51 KB JPEG file, contains logos, a color background, and the text "*Europass, certificate of participation, The European Digital Credentials for Learning Support Team certifies that Ana Andromeda has attended the EDC Updates webinar 12/02/2025*"

[17] The unsigned version of the joint degree has 11935 lines and 971 KB, it includes 51 base64 encoded PNG images, namely the logo of the Swedish Umeå Universitet 17 times, the logo of a hungarian learning provider 16 times, the logo of the National Agency for Quality Assurance in Higher Education 16 times, 1 logo of an internship in Ireland, 1 image of a sample mock credential joint degree, the images together weigth 414 KB. Removing the PNG files and compressing the JSON would make it become 63 KB, notice that the files attached to a PDF file are automatically stored in compressed format, thus as it is the compressed size that is relevant, removing the images would cause the 971 KB JSON file to occupy just 63 KB when stored inside the PDF. Comparing the signed (473 KB) and the unsigned (571 KB) versions of the Masters Degree sample file that has fewer images (33 PNG and 1 JPEG), without removing the images the signed JSON-LD file compressed around 5 to 1 and the unsigned JSON file compressed almost 8 to 1.

[18] `https://europa.eu/europass/elm-browser/documentation/rdf/ontology/documentation/index-en.html`

   (c) given name: `http://xmlns.com/foaf/spec/#term_givenName`
   (d) country of origin code: `http://data.europa.eu/snb/model/elm/countryCode`
   (e) European Student Identifier: `https://wiki.geant.org/display/SM/European+Student+Identifier`
   (f) sex / gender: `http://data.europa.eu/snb/model/elm/gender`
   (g) student email in the HEI: `http://data.europa.eu/snb/model/elm/emailAddress`

3. Course:
   (a) name of study programme: `http://data.europa.eu/snb/model/elm/LearningAchievementSpecification`
   (b) study programme (official) duration in ECTS credits: `http://data.europa.eu/snb/model/elm/creditPoint`
   (c) grade: `http://data.europa.eu/snb/model/elm/gradeStatus`
   (d) assessed by: `http://data.europa.eu/snb/model/elm/assessedBy`
   (e) grading scheme: `http://data.europa.eu/snb/model/elm/gradingScheme`
   (f) awarding date: `http://data.europa.eu/snb/model/elm/awardingDate`
   (g) ID verification: `http://data.europa.eu/snb/model/elm/idVerification`
   (h) EQF level: `http://data.europa.eu/snb/model/elm/EQFLevel`
   (i) thematic area: `http://data.europa.eu/snb/model/elm/ISCEDFCode`
   (j) entry requirements: `http://data.europa.eu/snb/model/elm/entryRequirement`
   (k) default language: `http://data.europa.eu/snb/model/elm/defaultLanguage`
   (l) landing page: `http://data.europa.eu/snb/model/elm/landingPage`
   (m) learning achievement specification: `http://data.europa.eu/snb/model/elm/learningAchievementSpecification`
   (n) result distribution: `http://data.europa.eu/snb/model/elm/ResultDistribution`
   (o) percentage lower: `http://data.europa.eu/snb/model/elm/percentageLower`
   (p) percentage equal: `http://data.europa.eu/snb/model/elm/percentageEqual`
   (q) percentage higher: `http://data.europa.eu/snb/model/elm/percentageHigher`
   (r) shortened grading: `http://data.europa.eu/snb/model/elm/ShortenedGrading`
   (s) contact hours: `http://data.europa.eu/snb/model/elm/contactHour`
   (t) volume of learning: `http://data.europa.eu/snb/model/elm/volumeOfLearning`
   (u) start date: `http://data.europa.eu/snb/model/elm/startDate`
   (v) end date: `http://data.europa.eu/snb/model/elm/endDate`
   (w) partial qualification: `http://data.europa.eu/snb/model/elm/isPartialQualification`
   (x) web resource: `http://data.europa.eu/snb/model/elm/WebResource`

4. VC:
   (a) verifiable credential: `http://data.europa.eu/snb/model/elm/verifiableCredential`
   (b) credential status: `http://data.europa.eu/snb/model/elm/CredentialStatus`
   (c) European Digital Credential: `http://data.europa.eu/snb/model/elm/EuropeanDigitalCredential`
   (d) proof: `http://data.europa.eu/snb/model/elm/Proof`
   (e) attachment: `http://data.europa.eu/snb/model/elm/attachment`
   (f) attachment type: `http://data.europa.eu/snb/model/elm/attachmentType`
   (g) credential subject: `http://data.europa.eu/snb/model/elm/subject`

5. Not found (fields that should exist but a reference for them was not found):
   (a) stackability (as defined in [Council of the European Union 2022, page 12])

1.6 Digital Wallets and User Onboarding

FIXME not yet described

1.7 Coping with Obsolescence and Technology Evolution

- To deliver the certificates use only PDF files stored in the web server file system together with a simple web app that can be easily replaced (used only to support searching for a certificate given its credential ID and reissuing or revoking certificates)
- Use PDFs that are as much as possible PDF/A (although with embedded files)
- Implement support for reissuing newer versions of the PDF certificates so that the embedded files can be updated

1.8 Technical Solution

The technical solution proposed here uses the following components:

1. An internet accessible virtual machine running Rocky Linux 9.x with NGINX (reverse proxy), Postgresql (in this case another database server could be used instead), Django, and the Django subapp `access-certificates` from the Django app `analog-digital-certificate` (web site to handle access to the certificates including handling certificate replacement and revocation)
2. A non internet accessible virtual machine running Rocky Linux 9.x with NGINX (reverse proxy), Postgresql, Django, and the two Django subapps `create-certificates` and `vc-interface` from the Django app `analog-digital-certificate` (application to help issuing certificates). This machine needs to interface with the DC4EU VC docker containers, they could be deployed in the same machine. It integrates with the HEI information system by being able to import Excel / CSV or JSON files.
3. The desktop version (Windows, macOS, Linux) of the Portuguese government `autenticação.gov.app` to bulk sign the PDF certificates

The software is implemented using literate programming [Knuth 1984]. As such the source code (this PDF) is a file that uses LaTeX [Lamport 1994] for the documentation chunks, PlantUML [PlantUML contributors 2025] for the class diagrams, and (mostly) python code that is automatically extracted from this document using the software tool dotNoweb [Sousa 2001], which is a modification of the noweb literate programming tool [Ramsey 1994].

1.9 Sample Python code to attach files to a PDF

Using the python module `pikepdf`[19] it is straightforward to attach one or more files inside a PDF file. Here is a complete example that accepts a PDF file as input and creates an output PDF file by attaching to the input PDF file one or more attachments (if the input PDF file already had attachments then those attachments are removed and will not be present in the output PDF file).

[19] `https://pypi.org/project/pikepdf/`, `https://github.com/pikepdf/pikepdf`

18 ⟨*version/certificates/attach-files-to-pdf.py* 18⟩≡
```
import pikepdf # requires 'pip3 install pikepdf'
from pathlib import Path
import shutil
import sys

def addAttachments2Pdf(inputPDF, attachmentsList):
   outputPDF = inputPDF
   thePDF = pikepdf.Pdf.open(inputPDF, allow_overwriting_input=True)
   for k in range(len(attachmentsList)):
      filespec = pikepdf.AttachedFileSpec.from_filepath(thePDF,
         Path(attachmentsList[k]), description="detailed info",
         relationship=pikepdf.Name.Data)
      thePDF.attachments[f"item{k}"] = filespec
   thePDF.save(outputPDF)

if len(sys.argv) < 4:
   print(f"{sys.argv[0]}: wrong usage, expected minimum of 3 arguments")
   print(f"Usage: python {sys.argv[0]} inputPDF outputPDF attachFile1 [attachFile2 ...]")
   print("- adds attachFile1 [attachFile2 ...] as attachments to inputPDF saving it as outputPDF")
   print("- WARNING: if inputPDF already had attached files they are not included in outputPDF")
   sys.exit(1)
inputPDF = sys.argv[1]
outputPDF = sys.argv[2]
shutil.copyfile(inputPDF, outputPDF)
addAttachments2Pdf(outputPDF, sys.argv[3:])
```
This code is written to file `version/certificates/attach-files-to-pdf.py`.

1.10 Sample Python code to extract attachments from PDF

Extract all attachments inside the `inputPDF` file into the `outputPath` directory.

19 ⟨*version/certificates/extract-attached-files-from-pdf.py* 19⟩≡
```
import pikepdf # requires 'pip3 install pikepdf'
import sys
import os
def extractAttachmentsFromPDF(inputPDF, outputPath, verbose=True):
   thePdf = pikepdf.Pdf.open(inputPDF)
   if len(thePdf.attachments) != 0:
      os.makedirs(outputPath, exist_ok=True)
      for oneAttachment in thePdf.attachments:
         attachmentFileName = thePdf.attachments[oneAttachment].get_all_filenames()["/F"].decode("utf-8")
         attachmentFileBytes = thePdf.attachments[oneAttachment].get_file().read_bytes()
         attachmentOutputPathName = f"{outputPath}/{attachmentFileName}"
         with open(attachmentOutputPathName, "wb") as f:
            f.write(attachmentFileBytes)
         if verbose:
            print(f"Extracted {attachmentOutputPathName} with {len(attachmentFileBytes)} bytes")
   elif verbose:
      print(f"File '{inputPDF}' has no attachments inside")
if len(sys.argv) != 3:
   print(f"{sys.argv[0]}: wrong usage, expected 1 argument")
   print(f"Usage: python {sys.argv[0]} inputPDF outputPath")
   print("- extract attached files from PDF into outputPath")
   sys.exit(1)
extractAttachmentsFromPDF(sys.argv[1], sys.argv[2], verbose=True)
```
This code is written to file `version/certificates/extract-attached-files-from-pdf.py`.

1.11 Sample Python code to use vector graphics in reportlab

The open source version of the `reportlab`[20] python module that we are going to use to create the PDF does not support vector graphics[21] (e.g. EPS), it only supports raster graphic formats (e.g. JPEG, PNG). That causes a certificate with a PNG logo to have a much larger size, specially if we want to use images with good print quality (e.g. 300 dpi). The 238 micro credential certificates issued in April 2025 had that problem, because they were using raster graphics each PDF had a size of 164 KB before signing and the logotypes used were medium quality to avoid increasing the file size too much, the file size increased to 412 KB after adding the qualified digital signature.

Fortunately there is a solution to this problem of not being able to use vector graphics in the logotypes when using `reportlab`. If we want to avoid using raster images then we can use the `svglib`[22] module to read an SVG vector image file and convert it to a reportlab drawing object. If we only have an EPS file of the logo then we can use e.g. the application Inkscape[23] to open the EPS file and save it as a SVG file. Here is a complete example that creates a PDF file with a scaled version of an SVG image placed between two paragraphs of random text.

20 ⟨*version/certificates/insert-vector-graphic-in-reportlab.py* 20⟩≡

```
import sys
from svglib.svglib import svg2rlg  # requires 'pip3 install svglib'
from reportlab.graphics.shapes import Drawing # requires 'pip3 install reportlab'
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.platypus import SimpleDocTemplate, Paragraph

def convertSvgToDrawing(path, scaleFactor):
   drawing = svg2rlg(path)
   sx = sy = scaleFactor
   drawing.width, drawing.height = drawing.minWidth() * sx, drawing.height * sy
   drawing.scale(sx, sy)
   drawing._showBoundary = False # True puts a box around the image
   return drawing

if len(sys.argv) != 3:
   print(f"{sys.argv[0]}: wrong usage, expected 2 arguments")
   print(f"Usage: python {sys.argv[0]} svgFile scaleFactor")
   print("- create sample PDF file that includes sample text and scaled SVG image")
   print("- E.g.: python {sys.argv[0]} foo.svg 0.5")
   sys.exit(1)
inputSVG = sys.argv[1]
scaleFactor = float(sys.argv[2])
styles = getSampleStyleSheet()
pdf_path = "tmp-sketch.pdf"
doc = SimpleDocTemplate(pdf_path)
drawing = convertSvgToDrawing(inputSVG, scaleFactor)
story = [
   Paragraph("Lorem ipsum!", styles['Normal']),
   drawing,
   Paragraph("Dolores sit amet, consectetur adipiscing elit.", styles['Normal'])
]
doc.build(story)
```

This code is written to file `version/certificates/insert-vector-graphic-in-reportlab.py`.

[20] `https://pypi.org/project/reportlab/`, `https://github.com/MrBitBucket/reportlab-mirror`

[21] According to the reportlab FAQ: *Can I use vector graphics in my PDFs? Yes and no. The open-source package doesn't do this. PageCatcher (a component of our commercial ReportLab PLUS package) allows you to easily incorporate any vector image by saving it as a PDF and then using it exactly as you would an image file, and Report Markup Language accepts PDF files along with JPG, GIF and PNG.*"

[22] `https://pypi.org/project/svglib/`, `https://github.com/deeplook/svglib`

[23] `https://inkscape.org`

1.12 Certificate Creation

WARNING: this is a draft section, some of its contents are not yet OK or not compatible with the rest of this document or include data that is going to be removed. For the time being it is better to skip reading it.

1. ImportData:
   (a) ImportStaticData: import static data (everything except grades, attendance rates, and program sheets) from HEI's information system using an Excel or JSON file with optional detached digital signature
   (b) ImportEditionsAttendanceRate: (optional) import in Excel or JSON format attendance rate for a given edition of a course at one or more school classes with optional detached digital signature
   (c) ImportProgramSheets: import in Excel or JSON format a course (or learning path) set of one or more curricular units *program sheets* with learning goals, syllabus, assessment, bibliography in PT and EN with optional detached digital signature
   (d) ImportGrades: import the grades in Excel or JSON format for one or more curricular unit editions, in JSON or Excel format with optional detached digital signature

   Using correlation codes supports reimporting the same or similar data without creating duplicated records and warning if incompatible data was found
2. ExportData: export all the digital platform data in a single Excel file with a set of related Excel sheets
3. Issuer: a HEI or a service that independently can issue certificates inside the HEI
4. Student: full name, date of birth, sex/gender, private email, student number
5. Course: name, study level, etc.
6. CurricularUnit: a study unit that has a grade that needs to go into the certificate
7. CurricularPlan: assigns a set of curricular units to a given course version including specifying the year and semester
8. Registration: the set of curricular units that the student is registered into and has a grade
9. Edition: a curricular plan and a start and end date
10. SchoolClass: A, B, C, ...
11. Grade: between 0 and 20, D, F,
12. IssueCertificatesForCourse: select a single course and a range of dates (to pick the set of relevant course editions) and issue all *complete* certificates that have not yet been issued
13. IssueCertificatesForStudent: issue all (*complete* or incomplete) certificates for that student that have not yet been issued
14. Certificate: one or more PDFs that supersede each other and the ability to search for them

In the models whenever translated names are needed use `field_name` for the name in the HEI's native language and use `field_name_en` for the English version of that name

1.13 Django app Analog-Digital-Certificate

The goal is to convert the existing Django app that UMAIA deployed in production in April 2025 to a reusable Django app [Lopatin 2020] called `analog-digital-certificate` that only implements the minimum needed funtionality (and that e.g. UMAIA will use and configure and extend as appropriate). In practice it should be converted into 3 subapps that are released together but that can be deployed together or separately:

- `create-certificates`: already exists for single course micro credentials, needs to be converted into a reusable Django app and needs to be expanded to support degree granting courses (that is, courses that have more than one curricular unit)
- `vc-interface`: interface with the DC4EU VCs (not yet implemented)
- `access-certificates`: already exists, needs to be converted into a reusable Django app

1.13.1 Sub-app Create Certificates

We assume the existing information system is responsible for ensuring the integrity of the data, including courses information, student data, student registration in individual courses and the student results, that is, the grades. For instance, we do not care if a student concluded this course by attending some curricular units in another course that were later credited into this course, or that the student did more optional curricular units than was really necessary. It is up to the existing information system to inform us (by exporting data that we can consume) that the student concluded the course and which curricular units to include in the course certificate. For that reason we only need nine Django models:

1. `IssuingEntity`: represents the HEI or a given service inside the HEI
2. `StudentLearner`: represents the student including student number, name, gender, date of birth and country of origin
3. `CurricularUnit`: represents a curricular unit including its code, name, number of ECTS, prerequisites, learning outcomes, syllabus, etc.
4. `CurricularUnitGrade`: a grade that a given student had at a given date in a given curricular unit. It optionally includes the attendance rate, if the attendance rate is filled-in then the grade itself may be left empty, but both fields cannot be simultaneously empty. The combination of the curricular unit code and the student number is unique, that is, it is not allowed for the same student to have two grades in the same curricular unit
5. `CurricularPlan`: a set of one or more curricular units that together represent an instance of a learning path. The Postgres specific Django `ArrayField` is used to hold this data, which is a list of tuples, each tuple with the curricular unit code, the year and semester it is taught, and whether it is optional
6. `LearningPath`: a course that includes one or more curricular units
7. `LearningPathEdition`: a learning path instance that started in a given date and ended in another date
8. `CourseCertificate`: associated with a student and a learning path edition. It is possible to issue a certificate for a student that has not yet finished all the curricular units in the learning path, the certificate will only contain the curricular units that have been completed. Because of the need to support optional curricular units and complex scenarios it is not possible[24] to

[24] It is possible to detect that if e.g. we have a 180 ECTS learning path and the student has only 175 ECTS then the student did not yet finish the learning path, but if the student has 180 ECTS but there are several optional curricular units then it is possible for the student not to have yet finished that learning path.

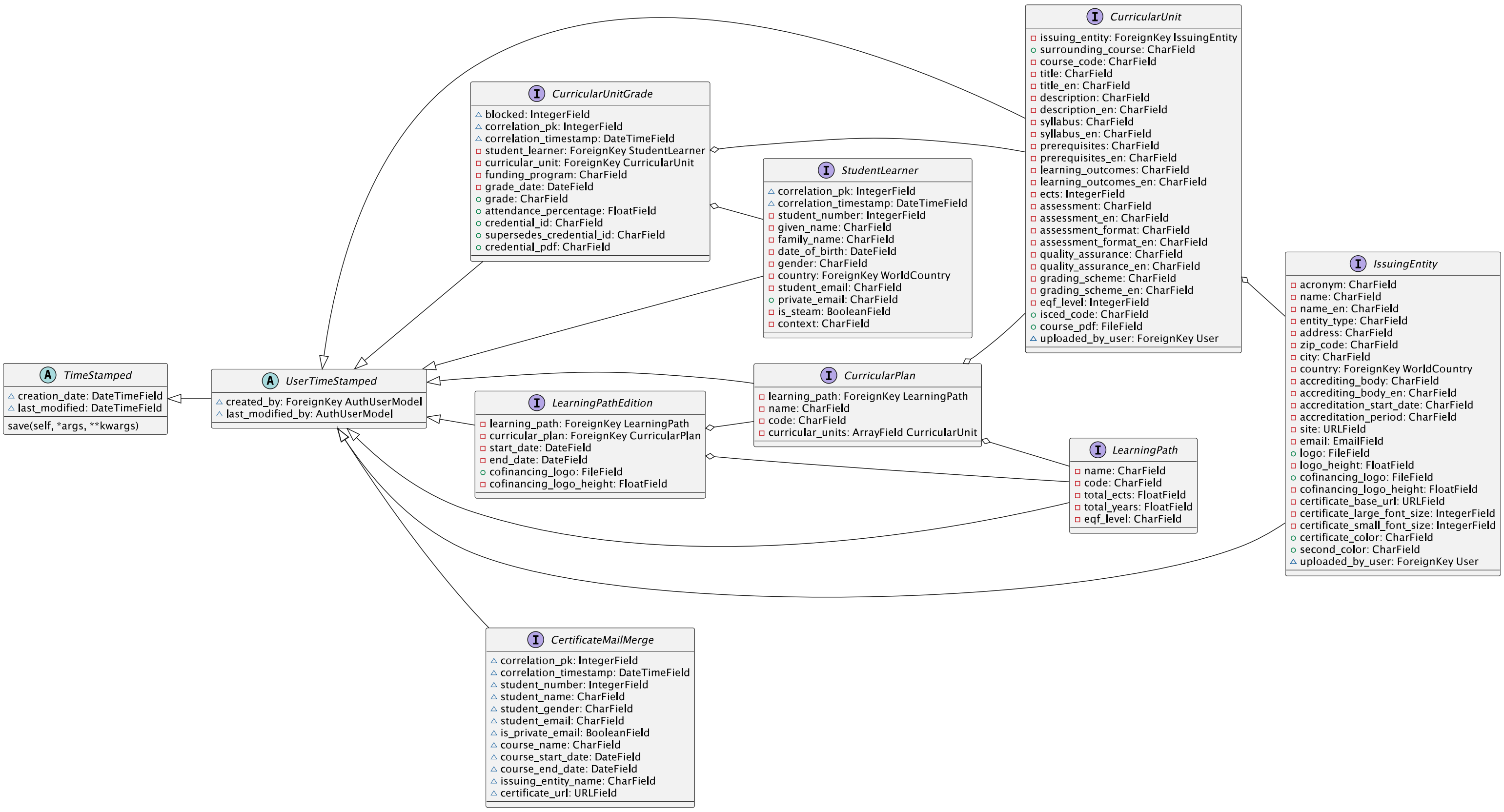

Figure 1.1: The Django models (FIXME: the CourseCertificate model is not yet included)

know if the student finished all the mandatory and optional curricular units in the learning path, it is up to the exported data obtained from the information system to specify that

9. `CertificateMailMerge`: used to be able to send an email to the student with a link to the certificate

In the case of an event certificate (as opposed to a course certificate) the StudentLearner is the person that attended the event, the curricular unit grade has an empty grade together with a non empty attendance rate (probably 100%), the CurricularUnit is the event itself, and the CurricularPlan includes only that curricular unit.

23 ⟨*version/certificates/models.py* 23⟩≡

```
# FIXME: this is a placeholder file, it still lacks some code and some imports
from django.contrib.postgres.fields import ArrayField
```

⟨*django model: class IssuingEntity* 25⟩
⟨*django model: class StudentLearner* 27⟩
⟨*django model: class CurricularUnit* 28⟩
⟨*django model: class CurricularUnitGrade* 34⟩
⟨*django model: class LearningPath* 31⟩
⟨*django model: class CurricularPlan* 32⟩
⟨*django model: class LearningPathEdition* 33⟩
⟨*django model: class CertificateMailMerge* 36⟩

This code is written to file `version/certificates/models.py`.

These two fields are used to support importing an Excel file, getting some errors, fixing those errors and reimporting again without creating duplicated objects. It is mostly important when a part of that Excel file was manually filled-in.

24 ⟨*fields correlation_pk and correlation_timestamp* 24⟩≡ (27 34 36)

```
correlation_pk = models.IntegerField(editable=False, null=False, blank=False, default=0,
        verbose_name=_("Correlation code"))
correlation_timestamp = models.DateTimeField(editable=False, null=True, blank=True, default=None,
        verbose_name=_("Correlation timestamp"))
```

Defines:
`correlation_pk`, used in chunks 27, 36, 42, and 45a.
`correlation_timestamp`, used in chunks 27, 36, 42, and 45a.

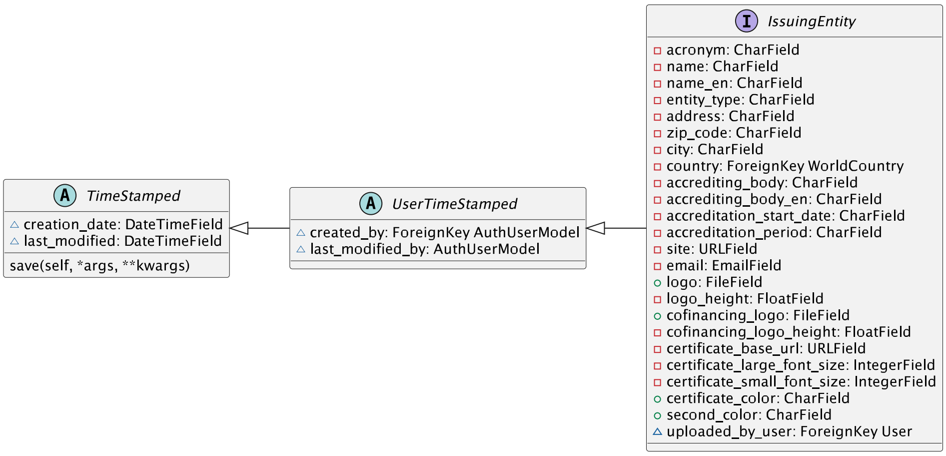

Figure 1.2: IssuingEntity

1.13.1.1 Django model IssuingEntity

25 ⟨*django model: class IssuingEntity* 25⟩≡ (23)

```
ENTITY_TYPE_CHOICES = (
   ("University", _("University")),
   ("Polytechnic", _("Polytechnic")),
   ("ProfessionalTraining", _("Professional Training")),
)

class IssuingEntity(UserTimeStamped):
   acronym = models.CharField(max_length=6, null=False, blank=False,
      verbose_name=_("Acronym"), unique=True)
   name = models.CharField(max_length=124, null=False, blank=False,
      verbose_name=_("Name (PT)"), unique=True)
   name_en = models.CharField(max_length=124, null=False, blank=False,
      verbose_name=_("Name (EN)"), unique=True)
   entity_type = models.CharField(max_length=11, choices=ENTITY_TYPE_CHOICES,
      blank=False, null=False, default="University",
      verbose_name=_("Entity type"))
   address = models.CharField(max_length=124, blank=False, null=False,
      verbose_name=_("Address"))
   zip_code = models.CharField(max_length=8, blank=False, null=False,
      verbose_name=_("Zip code"))
   city = models.CharField(max_length=11, blank=False, null=False,
      verbose_name=_("City"))
   country = models.ForeignKey(WorldCountry, null=False, blank=False,
      on_delete=models.RESTRICT, default=WorldCountry.get_default_country_pk,
      verbose_name=_("Country"))
   accrediting_body = models.CharField(max_length=100, blank=False, null=False,
      default="Agência de Avaliação e Acreditação do Ensino Superior (A3ES)",
      verbose_name=_("Accreditation Body (PT)"))
   accrediting_body_en = models.CharField(max_length=100, blank=False, null=False,
      default=(
         "Portuguese Agency for Assessment and Accreditation of Higher "
         "Education (A3ES)"
      ), verbose_name=_("Accreditation Body (EN)"))
   accreditation_start_date = models.DateField(null=False, blank=False,
      verbose_name=_("Accreditation start date"),
```

```
        help_text="https://si.a3es.pt/sia3es/page?stage=ConsultaPublicaProcesso&processID=719")
    accreditation_period = models.IntegerField(blank=False, null=False,
        validators=[ MaxValueValidator(10), MinValueValidator(1) ],
        verbose_name = _("Accreditation validity period"),
        help_text = "https://si.a3es.pt/sia3es/page?stage=ConsultaPublicaResultados&clearParams=true")
    site = models.URLField(max_length=200, null=False, blank=False)
    email = models.EmailField(max_length=254, null=False, blank=False)
    logo = models.FileField(blank=True, null=True,
        upload_to=uploadToPathAndRenameWithAuthor('certificate-template/%Y/%m/%d'),
        verbose_name=_('Upload logo'), help_text=_("Issuing entity Logotype (.png)")
    )
    logo_height = models.FloatField(null=False, blank=False, default=1.0,
        help_text=_("Logo height in inches, width will be scaled appropriatelly"))
    cofinancing_logo = models.FileField(blank=True, null=True,
        upload_to=uploadToPathAndRenameWithAuthor('certificate-template/%Y/%m/%d'),
        verbose_name=_('Upload cofinancing logo'),
        help_text=_("Cofinancing logo (.png), leave empty if not necessary")
    )
    cofinancing_logo_height = models.FloatField(null=False, blank=False,
        default=0.6, help_text="Not yet used, hard coded")
    certificate_base_url = models.URLField(max_length=200, null=False, blank=False,
        help_text=_("Permanent certificates base URL, e.g. https://certificados.umaia.pt"))
    certificate_large_font_size = models.IntegerField(null=False, blank=False,
        default=16)
    certificate_small_font_size = models.IntegerField(null=False, blank=False,
        default=12)
    # the certificate_color cannot be NULL but it can be an empty string meaning black
    certificate_color = models.CharField(_("Color"), blank=True, max_length=10,
        help_text=_("Issuing entity color to use in certificate"))
    second_color = models.CharField(_("Second color"), blank=True, max_length=10,
        help_text=_("Second color to use in certificate"))
    def colored_name(self):
        return format_html('<span style="color: {};">{}</span>',
                    self.certificate_color, self.certificate_color)
    colored_name.short_description = _("Certificate color")
    ⟨common django field: uploaded_by_user (never defined)⟩
    def __str__(self): # when displayed as foreign key
        return "%s - %s (accreditated %s for %d years)" % (self.acronym, self.name_pt,
            self.accreditation_start_date, self.accreditation_period)
    class Meta:
        verbose_name = _("M.01 Issuing Entity")
        verbose_name_plural = _("M.01 Issuing Entities")
```

Defines:
ENTITY_TYPE_CHOICES, never used.
IssuingEntity, used in chunks 14, 28, 40c, and 41a.

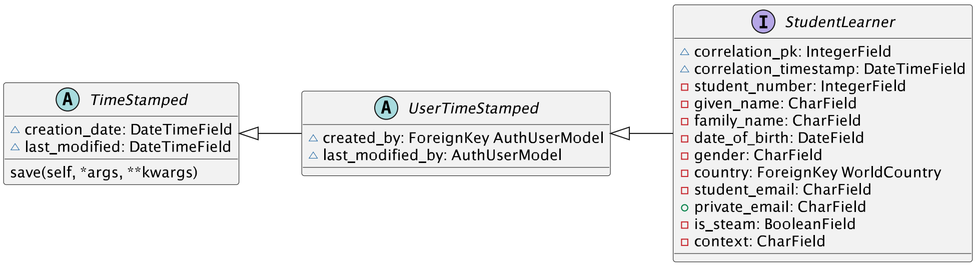

Figure 1.3: StudentLearner

1.13.1.2 Django model StudentLearner

27 ⟨*django model: class StudentLearner* 27⟩≡ (23)

```
LEARNER_GENDER_CHOICES = (
   ("F", "Female"),
   ("M", "Male"),
   ("U", "Unknown"),
)

class StudentLearner(UserTimeStamped):
   ⟨fields correlation_pk and correlation_timestamp 24⟩
   student_number = models.IntegerField(blank=False, null=False, unique=True,
      validators=[ MaxValueValidator(999_000), MinValueValidator(1) ],
      verbose_name = _("Student number"))
   given_name = models.CharField(max_length=100, null=False, blank=False,
      verbose_name=_("Given name"))
   family_name = models.CharField(max_length=100, null=False, blank=False,
      verbose_name=_("Family name"))
   date_of_birth = models.DateField(null=False, blank=False,
      verbose_name=_("Date of birth"))
   gender = models.CharField(max_length=1, choices=LEARNER_GENDER_CHOICES,
      blank=True, null=True, default="U", verbose_name=_("Gender"))
   country = models.ForeignKey(WorldCountry, null=False, blank=False,
      on_delete=models.RESTRICT, default=WorldCountry.get_default_country_pk,
      verbose_name=_("Country"))
   student_email = models.EmailField(max_length=254, null=False, blank=False)
   private_email = models.EmailField(max_length=254, null=True, blank=True)
   is_steam = models.BooleanField(null=True, blank=True,
      verbose_name=_("STEAM background"))
   context = models.CharField(max_length=50, null=False, blank=False, default="Unknown",
      verbose_name=_("Context"))
   def __str__(self): # when displayed as foreign key
      return "%d %s %s" % (self.student_number, self.given_name, self.family_name)
   class Meta:
      indexes = [
         models.Index(fields=["correlation_pk", "correlation_timestamp"]),
      ]
      verbose_name = _("M.05 Student Learner")
      verbose_name_plural = _("M.05 Student Learners")
```

Defines:
`LEARNER_GENDER_CHOICES`, never used.
`StudentLearner`, used in chunks 34 and 42.
Uses `correlation_pk` 24 and `correlation_timestamp` 24.

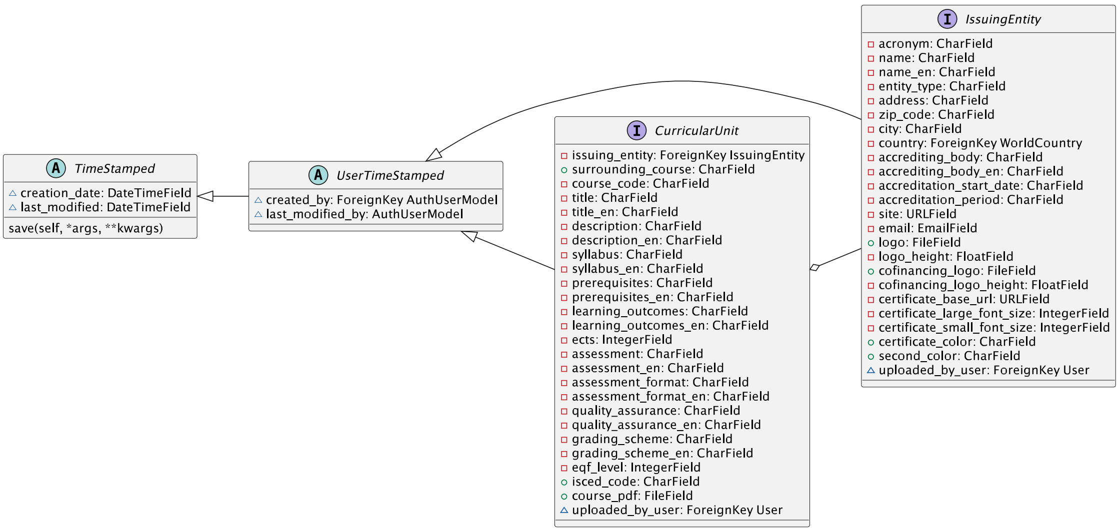

Figure 1.4: CurricularUnit

1.13.1.3 Django model CurricularUnit

28 ⟨*django model: class CurricularUnit 28*⟩≡ (23)

```
class CurricularUnit(UserTimeStamped):
    issuing_entity = models.ForeignKey(IssuingEntity, null=False, blank=False,
        on_delete=models.RESTRICT, verbose_name=_("Issuing entity"))
    surrounding_course = models.CharField(max_length=80, null=True, blank=True,
        verbose_name=_("Surrounding course (PT)"),
        help_text=_("Course that this curricular unit belongs to"))
    course_code = models.CharField(max_length=6, null=False, blank=False,
        verbose_name=_("Code"), help_text=_("Code used in the HEI information system"),
        unique=True)
    title = models.CharField(max_length=80, null=False, blank=False,
        verbose_name=_("Title (PT)"), unique=True)
    title_en = models.CharField(max_length=80, null=False, blank=False,
        verbose_name=_("Title (EN)"), unique=True)
    description = models.CharField(max_length=300, null=False, blank=False,
        verbose_name=_("Description (PT)"), default="FIXME description_pt")
    description_en = models.CharField(max_length=300, null=False, blank=False,
        verbose_name=_("Description (EN)"), default="FIXME description_en")
    syllabus = models.CharField(max_length=1000, null=False, blank=False,
        verbose_name=_("Syllabus (PT)"), default="FIXME syllabus_pt")
    syllabus_en = models.CharField(max_length=1000, null=False, blank=False,
        verbose_name=_("Syllabus (EN)"), default="FIXME syllabus_en")
    prerequisites = models.CharField(max_length=250, null=False, blank=False,
        verbose_name=_("Prerequisites (PT)"), default="FIXME prerequisites_pt")
    prerequisites_en = models.CharField(max_length=250, null=False, blank=False,
        verbose_name=_("Prerequisites (EN)"), default="FIXME prerequisites_en")
    learning_outcomes = models.CharField(max_length=1000, null=False, blank=False,
        verbose_name=_("Learning outcomes (PT)"),
        default="FIXME learning_outcomes_pt")
    learning_outcomes_en = models.CharField(max_length=1000, null=False, blank=False,
        verbose_name=_("Learning outcomes (EN)"),
        default="FIXME learning_outcomes_en")
    ects = models.IntegerField(blank=False, null=False,
        validators=[ MaxValueValidator(60), MinValueValidator(1) ],
```

```
        verbose_name = _("ECTS"))
    assessment = models.CharField(max_length=80, null=False, blank=False,
        verbose_name=_("Assessment (PT)"),
        default="Avaliação baseada na realização de projetos")
    assessment_en = models.CharField(max_length=80, null=False, blank=False,
        verbose_name=_("Assessment (EN)"), default="Project-based assessment")
    assessment_format = models.CharField(max_length=80, null=False, blank=False,
        verbose_name=_("Format(s) of assessment (PT)"),
        default="Classificação dada manualmente pelo docente")
    assessment_format_en = models.CharField(max_length=80, null=False, blank=False,
        verbose_name=_("Format(s) of assessment (EN)"),
        default="Manual grading by the teacher")
    quality_assurance = models.CharField(max_length=600, null=False, blank=False,
        verbose_name=_("Quality assurance (PT)"), default = (
            "Curso criado em colaboração com parceiros externos, de acordo com o "
            "regime jurídico do Decreto-Lei n.º 74/2006, de 24 de março, alterado "
            "pelos Decretos-Lei n.ºs 107/2008, de 25 de junho, 230/2009, de 14 de "
            "setembro, 115/2013, de 7 de agosto, pela Lei n.º 63/2016, de 13 de "
            "setembro, pelos Decretos-Lei n.ºs 65/2018, de 16 de agosto, e 27/2021, "
            "de 16 de abril. Unidade curricular submetida a análise pelo Conselho "
            "Científico e homologada pelo Reitor, serviço docente atribuído "
            "de acordo com as regras de seleção de docentes para lecionar unidades "
            "curriculares de licenciatura, mestrado ou doutoramento"
        ))
    quality_assurance_en = models.CharField(max_length=600, null=False, blank=False,
        verbose_name=_("Quality assurance (EN)"), default = (
            "Course created in collaboration with external partners, in accordance "
            "with the legal regime of Decree-Law No. 74/2006, of March 24, amended by "
            "Decree-Laws No. 107/2008, of June 25, 230/2009, of September 14, "
            "115/2013, of August 7, the Law 63/2016, of September 13, Decree-Laws No. "
            "65/2018, of August 16, and 27/2021, of April 16. Curricular unit submitted "
            "for analysis by the Scientific Council and approved "
            "by the Rector, teaching service assigned in accordance with the rules for "
            "selecting teachers to teach undergraduate or graduate courses"
        ))
    grading_scheme = models.CharField(max_length=150, null=False, blank=False,
        verbose_name=_("Grading scheme (PT)"),
        default="(D)esistiu, (F)altou à avaliação, Nota 0..20, 10 para passar")
    grading_scheme_en = models.CharField(max_length=150, null=False, blank=False,
        verbose_name=_("Grading scheme (EN)"),
        default="(D)ropped out, (F)ailed to attend assessment, Grade 0..20, 10 is pass grade")
    eqf_level = models.IntegerField(blank=False, null=False,
        validators=[ MaxValueValidator(8), MinValueValidator(4) ],
        verbose_name = _("EQF level"), default=5,
        help_text="https://en.wikipedia.org/wiki/European_Qualifications_Framework")
        # 5 The descriptor for the higher education short cycle (within or linked to
        #   the first cycle), developed by the Joint Quality Initiative as part of the
        #   Bologna process, corresponds to the learning outcomes for EQF level 5.
    isced_code = models.CharField(max_length=150, null=True, blank=True,
        verbose_name=_("ISCED-F 2013"),
        help_text=_(
            "ISCED Fields of Education and Training 2013: "
            "https://uis.unesco.org/en/topic/international-standard-classification-education-isced"
        ))
    course_pdf = models.FileField(
        upload_to=uploadToPathAndRenameWithAuthor('courses/%Y/%m/%d'),
        verbose_name=_('Course PDF'), null=True, blank=True,
        help_text=_("Upload PDF file with detailed information about the course")
```

```
    )
    ⟨common django field: uploaded_by_user (never defined)⟩
    def __str__(self): # when displayed as foreign key
        return "%s (%d ECTS)" % (self.title, self.ects)
    class Meta:
        verbose_name = _("M.02 Curricular Unit")
        verbose_name_plural = _("M.02 Curricular Unit")
```

Defines:
`CurricularUnit`, used in chunks 34 and 41–43.
Uses `IssuingEntity` 25.

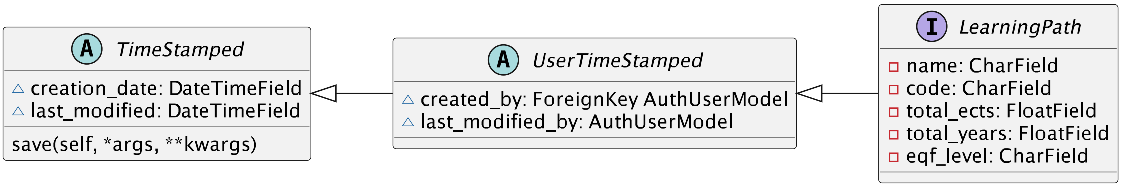

Figure 1.5: LearningPath

1.13.1.4 Django model LearningPath

31 ⟨*django model: class LearningPath* 31⟩≡ (23)

```
EQF_LEVEL_CHOICES = (
   ("Level1", _("1: Basic general knowledge")),
   ("Level2", _("2: Basic factual knowledge of a field of work or study")),
   ("Level3", _("3: Knowledge of facts, principles, processes and general concepts, "
      "in a field of work or study")),
   ("Level4", _("4: Factual and theoretical knowledge in broad contexts within a field "
      "of work or study")),
   ("Level5", _("5: Comprehensive, specialised, factual and theoretical knowledge "
      "within a field of work or study and awareness of boundaries of that knowledge")),
   ("Level6", _("6: Advanced knowledge of a field of work or study, involving a "
      "critical understanding of theories and principles")),
   ("Level7", _("7: Highly specialised knowledge, some of which is at the forefront of "
      "knowledge in a field of work or study")),
   ("Level8", _("8: PhD level")),
)

class LearningPath(UserTimeStamped):
   name = models.CharField(max_length=100, null=False, blank=False,
      verbose_name=_("Name"))
   code = models.CharField(max_length=100, null=False, blank=False,
      verbose_name=_("Code"))
   total_ects = models.FloatField(blank=False, null=False,
      validators=[ MaxValueValidator(360.0), MinValueValidator(0.1) ],
      verbose_name = _("Total ECTS"),
      help_text = _("Sum of the ECTS of the curricular units"))
   total_years = models.FloatField(blank=False, null=False,
      validators=[ MaxValueValidator(6.0), MinValueValidator(0.1) ],
      verbose_name = _("Years"),
      help_text = _("Total number of years to complete this learning path"))
   eqf_level = models.CharField(max_length=6, choices=EQF_LEVEL_CHOICES,
      blank=True, null=True, verbose_name=_("EQF level"))
```

Defines:
`EQF_LEVEL_CHOICES`, never used.
`LearningPath`, used in chunks 32, 33, 43, and 44a.

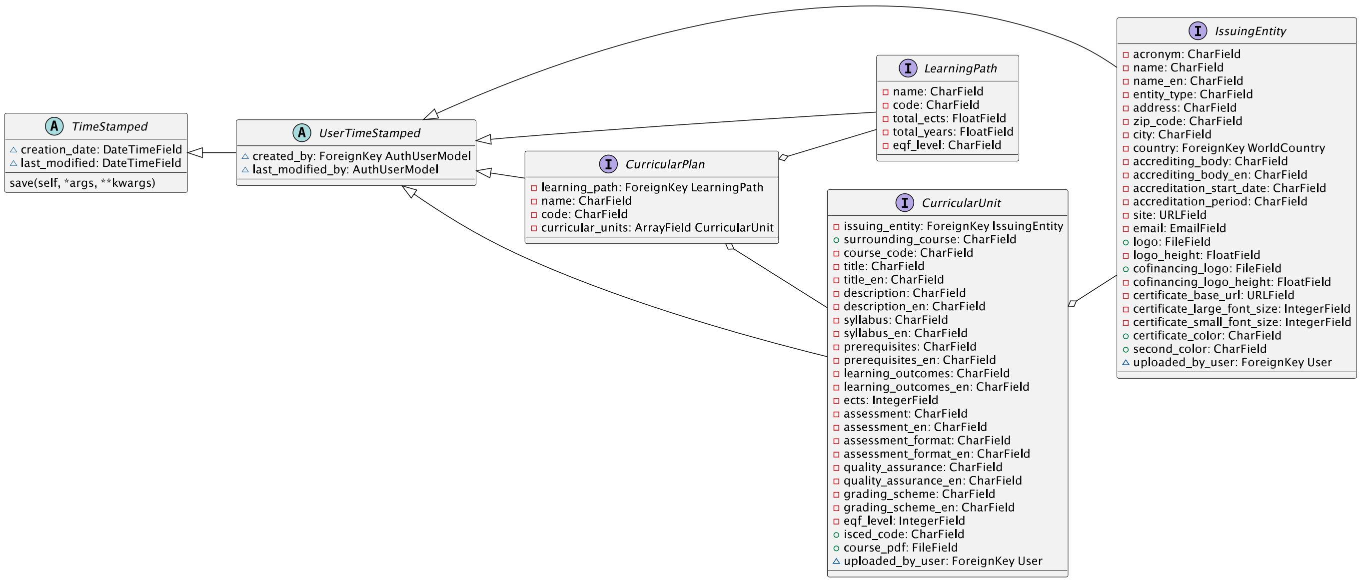

Figure 1.6: CurricularPlan

1.13.1.5 Django model CurricularPlan

This model requires a PostgreSQL database backend because it uses `ArrayField`. The `curricular_units` field will contain a list of 4-tuples, each 4-tuple with `(CU_code, year, semester, optional)`, specifying the code of the curricular unit, the year and the semester where that curricular unit is taught, and whether it is optional.

32 ⟨*django model: class CurricularPlan* 32⟩≡ (23)

```
class CurricularPlan(UserTimeStamped):
    learning_path = models.ForeignKey(LearningPath, null=False, blank=False,
        on_delete=models.RESTRICT, verbose_name=_("Learning path"))
    name = models.CharField(max_length=100, null=False, blank=False,
        verbose_name=_("Name"))
    code = models.CharField(max_length=100, null=False, blank=False,
        verbose_name=_("Code"))
    # this is a Postgres database specific field:
    curricular_units = ArrayField(ArrayField(models.CharField()))
```

Defines:
`CurricularPlan`, used in chunks 33, 43d, and 44a.
Uses `LearningPath` 31.

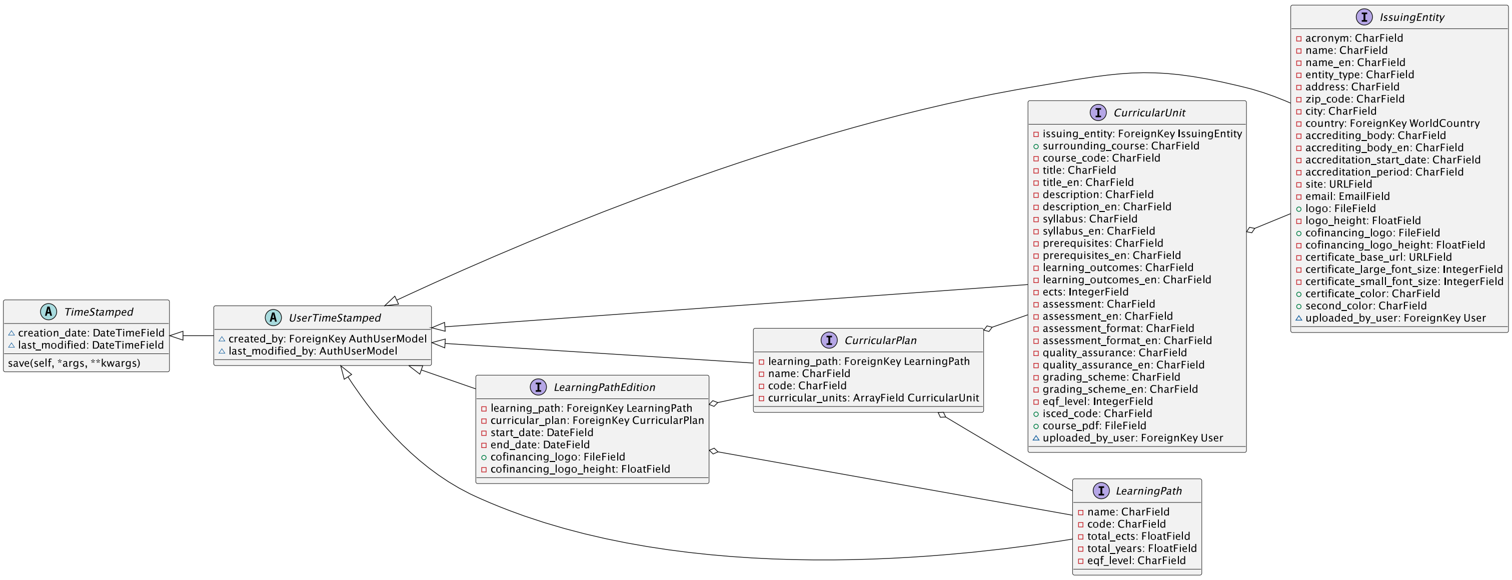

Figure 1.7: LearningPathEdition

1.13.1.6 Django model LearningPathEdition

33 ⟨*django model: class LearningPathEdition* 33⟩≡ (23)

```
class LearningPathEdition(UserTimeStamped):
    learning_path = models.ForeignKey(LearningPath, null=False, blank=False,
        on_delete=models.RESTRICT, verbose_name=_("Learning path"))
    curricular_plan = models.ForeignKey(CurricularPlan, null=False, blank=False,
        on_delete=models.RESTRICT, verbose_name=_("Curricular plan"))
    start_date = models.DateField(null=False, blank=False, verbose_name=_("Start date"))
    end_date = models.DateField(null=False, blank=False, verbose_name=_("End date"))
    cofinancing_logo = models.FileField(blank=True, null=True,
        upload_to=uploadToPathAndRenameWithAuthor('certificate-template/%Y/%m/%d'),
        verbose_name=_('Upload cofinancing logo'),
        help_text=_("Cofinancing logo (.png), leave empty if not necessary")
    )
    cofinancing_logo_height = models.FloatField(null=False, blank=False,
        default=0.6, help_text="Not yet used, hard coded")
```

Defines:
`LearningPathEdition`, used in chunk 44a.
Uses `CurricularPlan` 32 and `LearningPath` 31.

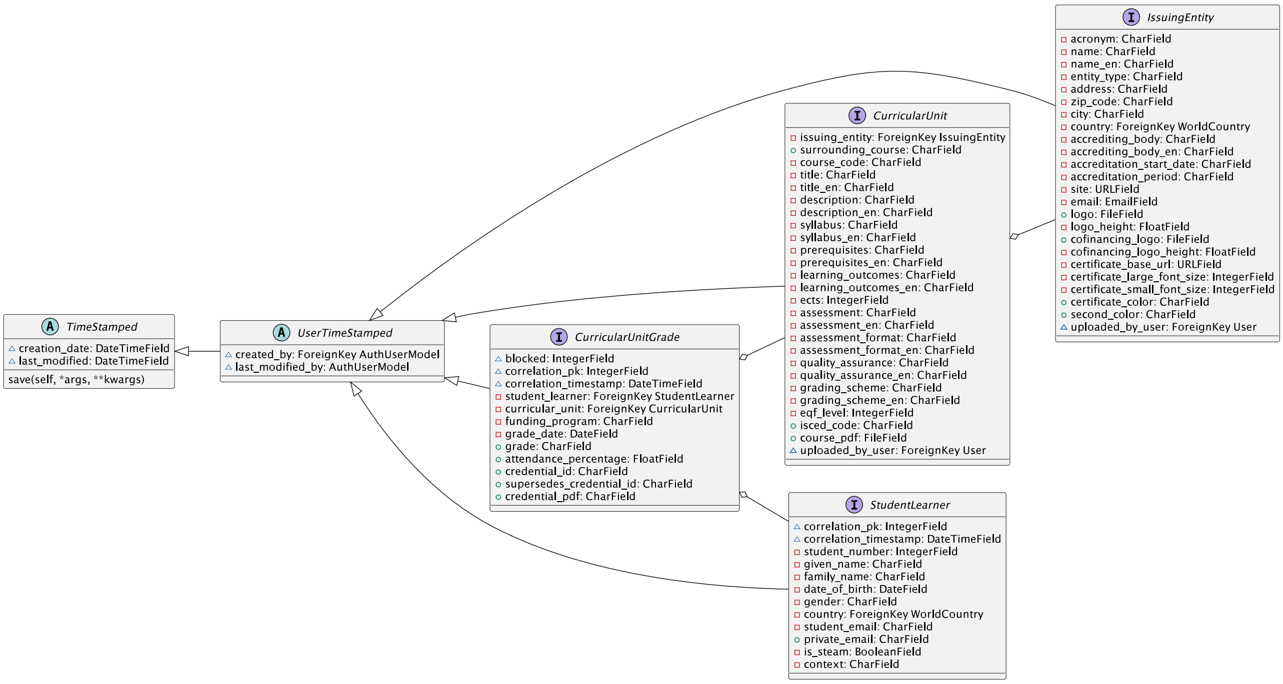

Figure 1.8: CurricularUnitGrade

1.13.1.7 Django model CurricularUnitGrade

34 ⟨*django model: class CurricularUnitGrade* 34⟩≡ (23)

```
class CurricularUnitGrade(UserTimeStamped):
   blocked = models.IntegerField(null=False, blank=False,
      default=0, verbose_name=_("Blocked by"), editable=False,
      help_text=_("If certificates generated no longer possible to change "
         "it unless the 'IssueCertificate' with this pk is deleted"))
   ⟨fields correlation_pk and correlation_timestamp 24⟩
   student_learner = models.ForeignKey(StudentLearner, null=False, blank=False,
      on_delete=models.RESTRICT, verbose_name=_("Student learner"), editable=False)
   curricular_unit = models.ForeignKey(CurricularUnit, null=False, blank=False,
      on_delete=models.RESTRICT, verbose_name=_("Course edition"), editable=False)
   funding_program = models.CharField(max_length=10, blank=False, null=False,
      choices=FUNDING_PROGRAM_TYPE_CHOICES, verbose_name=_("Funding program"),
      editable=False)
   grade_date = models.DateField(null=False, blank=False, verbose_name=_("Grade date"))
   grade = models.CharField(max_length=2, null=True, blank=True, editable=False,
      verbose_name=_("Grade"),
      help_text=( # Passar/Falhar / Pass/Fail
         "(D)esistiu, (F)altou, Nota 0..20, 10 para passar / "
         "(D)ropped out, (F)ailed to attend assessment, Grade 0..20, 10 is pass grade"
      ))
   attendance_percentage = models.FloatField(blank=True, null=True,
      validators=[ MaxValueValidator(1.0), MinValueValidator(0.0) ],
      verbose_name = _("Attendance percentage"), editable=False)
   credential_id = models.CharField(max_length=32, blank=True, null=True,editable=False)
   supersedes_credential_id = models.CharField(max_length=32, blank=True, null=True,editable=False)
   credential_pdf = models.CharField(null=True, blank=True, editable=False)
   def __str__(self): # when displayed as foreign key
      return "%s (%s, grade: %s, attendance: %.2f%%)" % (self.micro_learner,
         self.course_edition, self.grade, self.attendance_percentage * 100)
```

```
    class Meta:
        constraints = [
            models.UniqueConstraint(fields=['student_learner', 'curricular_unit', ],
                name='⟨only one record per learner per curricular unit 35⟩')
          ]
        indexes = [
            models.Index(fields=["credential_id"]),
        ]
        verbose_name = _("M.06 Curricular Unit Grade")
        verbose_name_plural = _("M.06 Curricular Unit Grades")
```

Defines:
`CurricularUnitGrade`, used in chunk 42c.
Uses `CurricularUnit` 28 and `StudentLearner` 27.

35 ⟨*only one record per learner per curricular unit* 35⟩≡ (34)

```
only_one_record_per_learner_per_curricular_unit
```

Defines:
`only_one_record_per_learner_per_curricular_unit`, never used.

Figure 1.9: CertificateMailMerge

1.13.1.8 Django model CertificateMailMerge

36 ⟨*django model: class CertificateMailMerge* 36⟩≡ (23)

```
class CertificateMailMerge(UserTimeStamped):
   ⟨fields correlation_pk and correlation_timestamp 24⟩
   student_number = models.IntegerField(blank=False, null=False, editable=False)
   student_name = models.CharField(max_length=200, null=False, blank=False, editable=False)

   student_gender = models.CharField(max_length=1, blank=False, null=False, editable=False)
   student_email = models.EmailField(max_length=254, null=False, blank=False, editable=False)
   is_private_email = models.BooleanField(blank=False, null=False, editable=False)
   course_name = models.CharField(max_length=200, null=False, blank=False, editable=False)
   course_start_date = models.DateField(null=False, blank=False, editable=False)
   course_end_date = models.DateField(null=False, blank=False, editable=False)
   issuing_entity_name = models.CharField(max_length=100, null=False, blank=False, editable=False)
   certificate_url = models.URLField(max_length=200, null=False, blank=False, editable=False)
   def __str__(self): # when displayed as foreign key
      return "%d %s %s" % (self.student_number, self.student_name, self.course_name)
   class Meta:
      indexes = [
         models.Index(fields=["correlation_pk", "correlation_timestamp"]),
      ]
      verbose_name = _("M.00 Certificate Mail Merge")
      verbose_name_plural = _("M.00 Certificate Mail Merge")
```

Defines:
`CertificateMailMerge`, used in chunk 45a.
Uses `correlation_pk` 24 and `correlation_timestamp` 24.

1.13.2 Sub-app Access Certificates

FIXME

1.13.3 Sub-app Verifiable Credentials Interface to the DC4EU Technical Stack

FIXME

1.14 Known Bugs

This document, besides still being incomplete, contains the following known bugs:

1. Section 1.12 needs to be fixed
2. The presented Django models are incomplete (e.g. they need customised code in the Django admin) and they still contain some default values that no longer make sense

1.15 Features of version deployed in April 2025

- April 2025 version 0.1: fully automated support for issuing Micro Credentials, signing them and delivering them (UMAIA issued 238 micro credential certificates, each with a single ad hoc JSON file embedded inside). Although in this case it took longer, the whole process, from obtaining the grades to sending the emails with the link to the signed certificate, could in principle be done in less than 1 hour
- To facilitate deployment the software was not implemented as a standalone separate Django App, it was integrated inside an already existing Django App, although it does not interact with the existing parts thus it is easy to extract
- The whole software is a single Django app. That is not a good idea security wise, because one part of the app should be available on the internet (access certificates), while the rest of the app should not (issue certificates). It needs to be broken into two subapps and added a third subapp (integrate with DC4EU VCs)

1.16 Roadmap

- April 2025:
  - Present the idea and the software implementation to ULUSOFONA and UP (done, 10 April 2025)
  - Remove known bugs from the Django app and upgrade it from using Django 4.2 LTS to using Django 5.2.1 LTS (done)
- May 2025
  - Fully document the idea so that it can be discussed with other partners (done, this document)
  - Expand this document with the source code of a Django reusable app (BEING DONE)
  - Discuss the possibility of joining forces with ULUSOFONA and UP
  - Prepare current code to be extracted from the Django app where it is being used and convert it into a reusable Django App, with 3 subapps that can be deployed together or separately:
    * Create certificates (code refactoring and migration BEING DONE)
    * Access certificates (code refactoring and migration BEING DONE)
    * Interface with VCs infrastructure from the DC4EU project (needs to be implemented from scratch)
  - Use `svglib` (see howto in Section 1.11) to add support for adding the logotypes in vector format (to decrease certificate size and to increase image quality)
  - UMAIA information system: add support for exporting all the *program sheets* in PT and EN of a given multi-year course in JSON format instead of in PDF
  - UMAIA information system: add support for exporting all the details of a student participation in a multi-year course in JSON format
  - Embed inside the PDF the markdown and the ELM v3 JSON-LD files

- June 2025
  - Test the DC4EU VCs infrastructure
- July 2025
  - Deploy into pre-production the DC4EU VCs infrastructure
  - Issue EducationalID[25] or MyAcademicID[26] credentials
- August 2025
  - Implement subapp that interfaces with the DC4EU VCs infrastructure
  - Implement support for issuing a certificate for a completed multi-year course
  - Issue a few multi-year course certificates (some of them with VCs embedded)
- September 2025
  - UMAIA information system: add support for adding detached digital signatures to the exported grades, the *program sheets*, and the details of a student participation in a multi-year course
  - Add support for reissuing certificates (wrong person name, wrong date of birth, etc.), revoking certificates (e.g. fraud detected) and from a `v1` certificate permanent link if it was reissued to obtain the latest version of that certificate
  - Add an API (using `djangorestframework`[27]) [Hillar 2018; Vincent 2022]
- October 2025
  - Issue certificate for Event participation
  - Issue certificate for Learning Path
  - Issue certificate for curricular units of an incomplete course
  - All certificates issued in this new format (PDF with files embedded inside), any student from any course can opt to use VCs
  - Deactivate the old way of issuing certificates

1.17 List of Acronyms

CSV — Comma Separated Values

DC4EU — Digital Credentials for Europe

DNS — Domain Name Service

ECTS — European Credit Transfer System

EBSI — European Blockchain Service Infrastructure

ELM v3 — European Learning Model version 3

EPS — Encapsulated PostScript (graphic file format)

HEI — Higher Education Institution

HTTP — Hyper Text Transfer Protocol

[25] `https://ec.europa.eu/digital-building-blocks/sites/display/EBSI/Educational+Identification+in+Europe`
[26] `https://myacademic-id.eu`
[27] `https://www.django-rest-framework.org`

HTTPS — Hyper Text Transfer Protocol Secure

JPEG — Joint Photographic Expert Group (graphic file format)

JSON — JavaScript Object Notation

JSON-LD — JavaScript Object Notation for Linked Data

LTS — Long Term Support

PDF — Portable Document Format

PDF/A — Portable Document Format / Archiving[28]

PKI — Public-Key Infrastructure

PNG — Portable Network Graphics (graphic file format)

ROI — Return On Investment

SVG — Scalable Vector Graphics (graphic file format)

UUID — Universally Unique IDentifier

VPN — Virtual Private Network

v1 — The first version of an issued certificate

v2 — The second version of a certificate if something needs to be corrected from the v1

VC — Verifiable Credential

1.18 Glossary

European Learning Model v3: (3 April 2023) a Data Model for Interoperability of Learning Opportunities, Qualifications, Accreditation and Credentials in Europe, developed by the European Commission. The European Learning Model is an extension of the W3C Verifiable Credentials Data Model expressed as JSON-LD, for the purposes of providing a standardised technical format of any learning described within the European Union and European Economic Area. `https://github.com/european-commission-empl/European-Learning-Model`

Verifiable Credential Data Model v2.0: W3C Proposed Recommendation of 20 March 2025, it is a specific way to express a set of claims made by an issuer, such as a driver's license or an education certificate. The verifiable credential specification describes the extensible data model for verifiable credentials, how they can be secured from tampering, and a three-party ecosystem for the exchange of these credentials that is composed of issuers, holders, and verifiers. `https://www.w3.org/TR/vc-data-model-2.0/`

1.19 Acknowledgments

This work was developed by the Portuguese education partners of the Digital Europe Project DC4EU — Digital Credentials for Europe, Grant Agreement no. 101102611 (April 2023 — July 2025), project coordinated by Ministerio de Asuntos Económicos y Transformación Digital (Spain), within the context of task WP5 (Large scale piloting of educational credentials and professional qualifications).

[28] Long-term preservation of electronic documents

1.20 Class diagrams

This section has the plantUML code used for creating the class diagrams.

40a ⟨*plantUML: abstract class TimeStamped* 40a⟩≡ (40–45)
```
abstract class TimeStamped {
   ~creation_date: DateTimeField
   ~last_modified: DateTimeField
   save(self, *args, **kwargs)
}
```

40b ⟨*plantUML: abstract class UserTimeStamped* 40b⟩≡ (40–45)
```
TimeStamped <|-- UserTimeStamped
abstract class UserTimeStamped {
   ~created_by: ForeignKey AuthUserModel
   ~last_modified_by: AuthUserModel
}
```

40c ⟨*plantUML: class IssuingEntity* 40c⟩≡ (40d 41b 43 44)
```
UserTimeStamped <|-- IssuingEntity
interface IssuingEntity {
   -acronym: CharField
   -name: CharField
   -name_en: CharField
   -entity_type: CharField
   -address: CharField
   -zip_code: CharField
   -city: CharField
   -country: ForeignKey WorldCountry
   -accrediting_body: CharField
   -accrediting_body_en: CharField
   -accreditation_start_date: CharField
   -accreditation_period: CharField
   -site: URLField
   -email: EmailField
   +logo: FileField
   -logo_height: FloatField
   +cofinancing_logo: FileField
   -cofinancing_logo_height: FloatField
   -certificate_base_url: URLField
   -certificate_large_font_size: IntegerField
   -certificate_small_font_size: IntegerField
   +certificate_color: CharField
   +second_color: CharField
   ~uploaded_by_user: ForeignKey User
}
```
Uses `IssuingEntity` 25.

40d ⟨*version/certificates/certificates-IssuingEntity.puml* 40d⟩≡
```
@startuml
left to right direction
⟨plantUML: abstract class TimeStamped 40a⟩
⟨plantUML: abstract class UserTimeStamped 40b⟩
⟨plantUML: class IssuingEntity 40c⟩
@enduml
```
This code is written to file `version/certificates/certificates-IssuingEntity.puml`.

41a ⟨*plantUML: class CurricularUnit* 41a⟩≡ (41b 43 44)
```
UserTimeStamped <|-- CurricularUnit
CurricularUnit o-- IssuingEntity
interface CurricularUnit {
   -issuing_entity: ForeignKey IssuingEntity
   +surrounding_course: CharField
   -course_code: CharField
   -title: CharField
   -title_en: CharField
   -description: CharField
   -description_en: CharField
   -syllabus: CharField
   -syllabus_en: CharField
   -prerequisites: CharField
   -prerequisites_en: CharField
   -learning_outcomes: CharField
   -learning_outcomes_en: CharField
   -ects: IntegerField
   -assessment: CharField
   -assessment_en: CharField
   -assessment_format: CharField
   -assessment_format_en: CharField
   -quality_assurance: CharField
   -quality_assurance_en: CharField
   -grading_scheme: CharField
   -grading_scheme_en: CharField
   -eqf_level: IntegerField
   +isced_code: CharField
   +course_pdf: FileField
   ~uploaded_by_user: ForeignKey User
}
```
Uses `CurricularUnit` 28 and `IssuingEntity` 25.

41b ⟨*version/certificates/certificates-CurricularUnit.puml* 41b⟩≡
```
@startuml
left to right direction
⟨plantUML: abstract class TimeStamped 40a⟩
⟨plantUML: abstract class UserTimeStamped 40b⟩
⟨plantUML: class IssuingEntity 40c⟩
⟨plantUML: class CurricularUnit 41a⟩
@enduml
```
This code is written to file `version/certificates/certificates-CurricularUnit.puml`.

42a ⟨*plantUML: class StudentLearner* 42a⟩≡ (42–44)

```
UserTimeStamped <|-- StudentLearner
interface StudentLearner {
   ~correlation_pk: IntegerField
   ~correlation_timestamp: DateTimeField
   -student_number: IntegerField
   -given_name: CharField
   -family_name: CharField
   -date_of_birth: DateField
   -gender: CharField
   -country: ForeignKey WorldCountry
   -student_email: CharField
   +private_email: CharField
   -is_steam: BooleanField
   -context: CharField
}
```

Uses `correlation_pk` 24, `correlation_timestamp` 24, and `StudentLearner` 27.

42b ⟨*version/certificates/certificates-StudentLearner.puml* 42b⟩≡

```
@startuml
left to right direction
⟨plantUML: abstract class TimeStamped 40a⟩
⟨plantUML: abstract class UserTimeStamped 40b⟩
⟨plantUML: class StudentLearner 42a⟩
@enduml
```

This code is written to file `version/certificates/certificates-StudentLearner.puml`.

42c ⟨*plantUML: class CurricularUnitGrade* 42c⟩≡ (43a 44c)

```
UserTimeStamped <|-- CurricularUnitGrade
CurricularUnitGrade o-- StudentLearner
CurricularUnitGrade o-- CurricularUnit
interface CurricularUnitGrade {
   ~blocked: IntegerField
   ~correlation_pk: IntegerField
   ~correlation_timestamp: DateTimeField
   -student_learner: ForeignKey StudentLearner
   -curricular_unit: ForeignKey CurricularUnit
   -funding_program: CharField
   -grade_date: DateField
   +grade: CharField
   +attendance_percentage: FloatField
   +credential_id: CharField
   +supersedes_credential_id: CharField
   +credential_pdf: CharField
}
```

Uses `correlation_pk` 24, `correlation_timestamp` 24, `CurricularUnit` 28, `CurricularUnitGrade` 34, and `StudentLearner` 27.

43a ⟨*version/certificates/certificates-CurricularUnitGrade.puml* 43a⟩≡
```
@startuml
left to right direction
⟨plantUML: abstract class TimeStamped 40a⟩
⟨plantUML: abstract class UserTimeStamped 40b⟩
⟨plantUML: class IssuingEntity 40c⟩
⟨plantUML: class CurricularUnit 41a⟩
⟨plantUML: class StudentLearner 42a⟩
⟨plantUML: class CurricularUnitGrade 42c⟩
@enduml
```
This code is written to file `version/certificates/certificates-CurricularUnitGrade.puml`.

43b ⟨*plantUML: class LearningPath* 43b⟩≡ (43 44)
```
UserTimeStamped <|-- LearningPath
interface LearningPath {
   -name: CharField
   -code: CharField
   -total_ects: FloatField
   -total_years: FloatField
   -eqf_level: CharField
}
```
Uses `LearningPath` 31.

43c ⟨*version/certificates/certificates-LearningPath.puml* 43c⟩≡
```
@startuml
left to right direction
⟨plantUML: abstract class TimeStamped 40a⟩
⟨plantUML: abstract class UserTimeStamped 40b⟩
⟨plantUML: class LearningPath 43b⟩
@enduml
```
This code is written to file `version/certificates/certificates-LearningPath.puml`.

43d ⟨*plantUML: class CurricularPlan* 43d⟩≡ (43 44)
```
UserTimeStamped <|-- CurricularPlan
CurricularPlan o-- LearningPath
CurricularPlan o-- CurricularUnit
interface CurricularPlan {
   -learning_path: ForeignKey LearningPath
   -name: CharField
   -code: CharField
   -curricular_units: ArrayField CurricularUnit
}
```
Uses `CurricularPlan` 32, `CurricularUnit` 28, and `LearningPath` 31.

43e ⟨*version/certificates/certificates-CurricularPlan.puml* 43e⟩≡
```
@startuml
left to right direction
⟨plantUML: abstract class TimeStamped 40a⟩
⟨plantUML: abstract class UserTimeStamped 40b⟩
⟨plantUML: class IssuingEntity 40c⟩
⟨plantUML: class LearningPath 43b⟩
⟨plantUML: class CurricularUnit 41a⟩
⟨plantUML: class CurricularPlan 43d⟩
@enduml
```
This code is written to file `version/certificates/certificates-CurricularPlan.puml`.

44a ⟨*plantUML: class LearningPathEdition* 44a⟩≡ (44)

```
UserTimeStamped <|-- LearningPathEdition
LearningPathEdition o-- LearningPath
LearningPathEdition o-- CurricularPlan
interface LearningPathEdition {
   -learning_path: ForeignKey LearningPath
   -curricular_plan: ForeignKey CurricularPlan
   -start_date: DateField
   -end_date: DateField
   +cofinancing_logo: FileField
   -cofinancing_logo_height: FloatField
}
```

Uses `CurricularPlan` 32, `LearningPath` 31, and `LearningPathEdition` 33.

44b ⟨*version/certificates/certificates-LearningPathEdition.puml* 44b⟩≡

```
@startuml
left to right direction
⟨plantUML: abstract class TimeStamped 40a⟩
⟨plantUML: abstract class UserTimeStamped 40b⟩
⟨plantUML: class IssuingEntity 40c⟩
⟨plantUML: class LearningPath 43b⟩
⟨plantUML: class CurricularUnit 41a⟩
⟨plantUML: class CurricularPlan 43d⟩
⟨plantUML: class LearningPathEdition 44a⟩
@enduml
```

This code is written to file `version/certificates/certificates-LearningPathEdition.puml`.

44c ⟨*version/certificates/certificates-AllModels.puml* 44c⟩≡

```
@startuml
left to right direction
⟨plantUML: abstract class TimeStamped 40a⟩
⟨plantUML: abstract class UserTimeStamped 40b⟩
⟨plantUML: class IssuingEntity 40c⟩
⟨plantUML: class StudentLearner 42a⟩
⟨plantUML: class CurricularUnit 41a⟩
⟨plantUML: class LearningPath 43b⟩
⟨plantUML: class CurricularPlan 43d⟩
⟨plantUML: class LearningPathEdition 44a⟩
⟨plantUML: class CurricularUnitGrade 42c⟩
⟨plantUML: class CertificateMailMerge 45a⟩
@enduml
```

This code is written to file `version/certificates/certificates-AllModels.puml`.

45a ⟨*plantUML: class CertificateMailMerge* 45a⟩≡ (44c 45b)

```
UserTimeStamped <|-- CertificateMailMerge
interface CertificateMailMerge {
   ~correlation_pk: IntegerField
   ~correlation_timestamp: DateTimeField
   ~student_number: IntegerField
   ~student_name: CharField
   ~student_gender: CharField
   ~student_email: CharField
   ~is_private_email: BooleanField
   ~course_name: CharField
   ~course_start_date: DateField
   ~course_end_date: DateField
   ~issuing_entity_name: CharField
   ~certificate_url: URLField
}
```

Uses `CertificateMailMerge` 36, `correlation_pk` 24, and `correlation_timestamp` 24.

45b ⟨*version/certificates/certificates-CertificateMailMerge.puml* 45b⟩≡

```
@startuml
left to right direction
⟨plantUML: abstract class TimeStamped 40a⟩
⟨plantUML: abstract class UserTimeStamped 40b⟩
⟨plantUML: class CertificateMailMerge 45a⟩
@enduml
```

This code is written to file `version/certificates/certificates-CertificateMailMerge.puml`.

1.21 Makefile

Used to process the code extracted from this file in the context of using tools available in different operating systems. Only relevant when using the literate programming system.

46 ⟨*version/certificates/Makefile* 46⟩≡

```
VERSION=$State: current $

include $(DOTNOWEB_LIB)/binRules$(OS)

WINDOWS_DRIVE=⟨Mac OS X WinXP mapped drive (never defined)⟩
INSTALLDIR=$(WINDOWS_DRIVE)/⟨Mac OS X install dir (never defined)⟩

local:

extern: ../../certificates.tid $(OBJDIR)

unitTests:
   @echo "***"; echo "*** unitTests of certificates.nw not yet built"

integrationTests:
   @echo "***"; echo "*** integrationTests of certificates.nw not yet built"

clean:
   rm -f *.pyc $(OBJDIR)/*

veryclean: clean
   rm -f *.py *.sh

install: extern
   @for f in attach-files-to-pdf.py extract-attached-files-from-pdf.py \
            insert-vector-graphic-in-reportlab.py; do \
      if older $$f $(INSTALLDIR)/$$f; then : ; else \
         if cp -fp $$f $(INSTALLDIR)/$$f; then \
               echo "Updated $(INSTALLDIR)/$$f"; else \
               echo "Error copying $$f, aborting"; \
               exit 1; \
         fi; \
      fi; done
   @find . -type f -name "*.puml" | while read f; do \
      if older $$f $(INSTALLDIR)/$$f; then : ; else \
         if cp -fp $$f $(INSTALLDIR)/$$f; then \
               echo "Updated $(INSTALLDIR)/$$f"; else \
               echo "Error copying $$f, aborting"; \
               exit 1; \
         fi; \
      fi; done
   @find ad_certificate -type f -name "*.py" | while read f; do \
      if older $$f $(INSTALLDIR)/django-ad-certificate/$$f; then : ; else \
         d=`dirname $$f`; \
         if [ ! -d $(INSTALLDIR)/django-ad-certificate/$$d ]; then \
            mkdir -p $(INSTALLDIR)/django-ad-certificate/$$d; \
      echo "Dir $(INSTALLDIR)/django-ad-certificate/$$d created"; fi; \
         if cp -p $$f $(INSTALLDIR)/django-ad-certificate/$$f; then \
               echo "Updated $(INSTALLDIR)/django-ad-certificate/$$f"; else \
               echo "Error copying $$f, aborting"; \
               exit 1; \
         fi; \
      fi; done
```

```
    @if older pyproject.toml $(INSTALLDIR)/django-ad-certificate/pyproject.toml; then \
       : ; else echo "Updating $(INSTALLDIR)/django-ad-certificate/pyproject.toml"; \
       cp -p pyproject.toml $(INSTALLDIR)/django-ad-certificate; fi

# the end
```

This code is written to file `version/certificates/Makefile`.

1.22 History / Changelog

v1.74 2025-05-08 First version that was distributed

v1.85 2025-05-12 Very minor text improvements and text fixes in several places; improved Section 1.3.8 reissuing and revoking certificates; improved Section 1.16 roadmap

v1.86 2025-07-22 Very minor text improvements and roadmap update

1.23 Code Chunks Index

1.24 Code Index

1.25 References

Arms, Caroline R. et al. (2014). The benefits and risks of the PDF/A-3 file format for archival institutions: an NDSA report. Technical report, National Digital Stewardship Alliance. http://hdl.loc.gov/loc.gdc/lcpub.2013655115.1.

Council of the European Union (2022, 16 December). Council Recommendation of 16 June 2022 on a European approach to micro-credentials for lifelong learning and employability (2022/C 243/02). `https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX%3A32022H0627%2802%29`.

Hillar, Gaston C. (2018). *Django RESTful Web Services*. Packt. ISBN 978-1-78883-392-9.

ISO 19005-3:2012(E) (2012). Document management — Electronic document file format for long-term preservation — Part 3: Use of ISO 32000-1 with support for embedded files (PDF/A-3). Standard, International Organization for Standardization, Geneva, CH.

ISO 32000-1:2008(E) (2008). Document management — Portable document format — Part 1: PDF 1.7. Standard, International Organization for Standardization, Geneva, CH. `https://opensource.adobe.com/dc-acrobat-sdk-docs/pdfstandards/PDF32000_2008.pdf`.

Knuth, Donald Ervin (1984). Literate programming. *The Computer Journal 27*(2), 97–111. http://www.literateprogramming.com/knuthweb.pdf.

Lamport, Leslie (1994). *LaTeX A Document Preparation System* (second ed.). Addison-Wesley. ISBN 0-201-52983-1.

Lopatin, Ben (2020). *Django Standalone Apps*. Apress. ISBN 978-1-4842-5631-2.

PlantUML contributors (2025, February 2). *Drawing UML with PlantUML — PlantUML Language Reference Guide*. Version 1.2025.0. https://plantuml.com/guide, https://github.com/plantuml/plantuml.

Portaria194/ (2021, 17 of September). Portaria n.º 194/2021 de 17 de setembro — define os modelos de diplomas e de certificados em formato eletrónico das ofertas educativas e formativas do ensino básico e secundário. https://diariodarepublica.pt/dr/detalhe/portaria/194-2021-171516470.

Preukschat, Alex and Drummond Reed (2021). *Self-Sovereign Identity: Decentralized digital identity and verifiable credentials*. Manning. ISBN 978-1-61729-659-8.

Ramsey, Norman (1994, September). Literate programming simplified. *IEEE Software 11*(5), 97–105. https://www.cs.tufts.edu/˜nr/noweb.

Sousa, Alexandre (2001, November). Literate programming applied to software maintenance and teaching computer science. *Perspectivas XXI 4*(8), 101–120. ISSN 0874-274X.

Vincent, William S. (2022, March 24). *Django 4.0 for APIs — Build web APIs with Python and Django*. Leanpub. http://leanpub.com/djangoforapis.

1.26 File Index

1.27 Subject Index