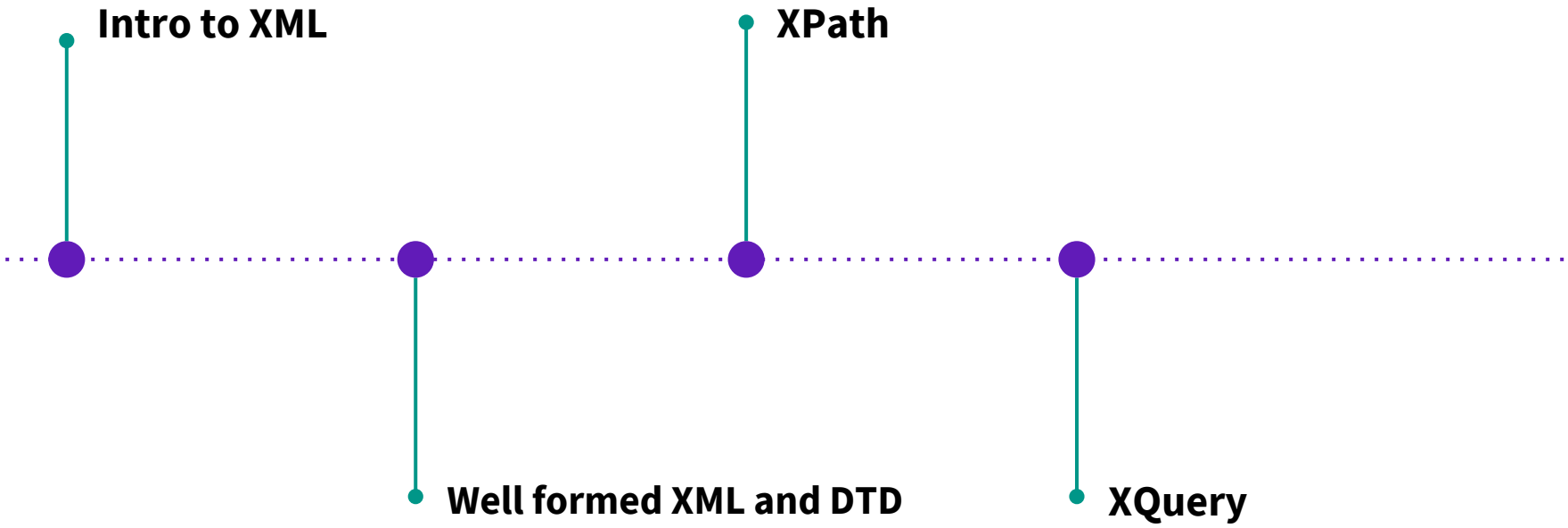
XML, XQuery and Xpath

Massimiliano Luca
massimiliano.luca@unitn.it

Nicolò Alessandro Girardini
nicolo.girardini@studenti.unitn.it

Introduction to Service Design and Engineering
Tech Lab: November 2018

Agenda



Tools

XML & XSD

- Text editor (Atom or Sublime are good)
- Free Formatter - XML XSD Validator (freeformatter.com/xml-validator-xsd.html)

XPath

- Free Formatter - XPath Tester (freeformatter.com/xpath-tester.html)

XQuery

- XQuery Tester (www.xpathtester.com/xquery)

Introduction to XML and XDSs

eXtensible Markup Language

- Standard for data representation and data exchanging
- Document format is similar to HTML
    - Tags describe contents instead of formatting

eXtensible Markup Language

```xml
<Universities>
  <University name="University of Trento" longitude="42.64" latitude="68.93">
    <Courses>
      <Course name="Knowledge and Data Integration" code="145324">
        <TeachingTeam>
          <Professor>
            <FirstName>Fausto</FirstName>
            <LastName>Giunchiglia</LastName>
          </Professor>
          <Assistant>
            <FirstName>Mattia</FirstName>
            <LastName>Fumagalli</LastName>
            <Title>Ph.D.</Title>
          </Assistant>
        </TeachingTeam>
      </Course>
    </Courses>
  </University>
</Universities>
```

Basic constructs

- Tagged elements
- Attributes
- Text

Structural requirements

- Single root
- DTD compliancy (if DTD is defined)
- Unique attributes within elements

eXtensible Markup Language

- To properly show XML as HTML use:
  - CSS (Cascading Stylesheets)
  - XSL (eXtensible Stylesheet Language)

eXtensible Markup Language

	Relational schema	XML
Structure	Table	Hierarchical Tree
Schema	Fixed in advance (ER / DB structure)	Flexible
Query	Simple human readable languages (SQL)	There are languages but are not that readable and simple
Ordering	None	Implied
Implementation	Native	Add-On

eXtensible Markup Language

- The Professor want to store some simple information to track who took this class (student id and name) and the mark of each student
- You are a web developer and you are developing a website for a big company. You want to store news, events, open positions, ...

eXtensible Markup Language

- The Professor want to store some simple information to track who took this class (student id and name) and the mark of each student

  A relation db: in this case the format of the data does not change so it is preferable to have a schema which is well defined

- You are a web developer and you are developing a website for a big company. You want to store news, events, open positions, ...

  XML: it is a more dynamic environment and a relational schema is to fixed to model this scenario

XSDs

- Focus on Document Type Descriptor
- Grammar to specify the elements, attributes, nesting, ordering, number of occurrences, IDs and IDREFs
- IDs and IDREFs are nothing more than pointers

XSDs

Basic components:

example

NOTE: With XSD also data types can be specified

XSDs

With Schema	Without Schema
● Programmers that want to use the resource can assume a structure ● If there is a structure, XSL or CSS can be used to enhance the visualization ● There is some kind of specification and so the exchange of data can be simpler to implement	● Flexibility ● XSDs may contain errors and as far as XSDs are used to valid XML files this is a huge problem (a.k.a. XSDs can be messy) ● Same for DTDs

Practice

Exercise 1

Build an XML file with those characteristics:

- `students` is the root node
- There will be two `student` subnodes
- Each student node will have as attributes: `id`, `name`, `age`, `major` (course path)
- Each student will have a `results` subnode: `result` will be the name of the subelement, with attributes `course` and `grade` (optional)
- The first student is: 198449, Massimiliano, 23, Computer Science. He has taken the Math course with grade C-
- The second student is: 203265, Nicolò, 23, Computer Science. He took the Math course with grade A and the Data Analysis course, but we don’t know the grade

Exercise 1

```
<students>
  <student id="198449" name="Massimiliano" age="22" major="Computer Science">
    <results>
      <result course="Math" grade="C-"/>
    </results>
  </student>
  <student id="203265" name="Nicolò" age="23" major="Computer Science">
    <results>
      <result course="Math" grade="A"/>
      <result course="Data Analysis"/>
    </results>
  </student>
</students>
```

Exercise 2

Working on the previous XML file:

- `students` has to become a more structured node
- Leave `id` as attribute and `results` as it is
- Each student node will have the other attributes as subnodes

Exercise 2

```xml
<students>
  <student id="198449">
    <name>Massimiliano</name>
    <age>22</age>
    <major>Computer Science</major>
    <results>
      <result course="Math" grade="C-"/>
    </results>
  </student>
  <student id="192414">
    <name>Nicolo'</name>
    <major>Computer Science</major>
    <major>Secutiy and Network</major>
    <results>
      <result course="Math" grade="A"/>
      <result course="Data Analysis"/>
    </results>
  </student>
</students>
```

Exercise 3

Given the solution to the

previous exercise, provide

a possible XSD

```
<students>
  <student name="David"></student>
  <student name="Craig"></student>
  <student name="Erik"></student>
</students>
```

Exercise 3

Given the solution to the previous exercise, provide a possible XSD

```xml
<students>
  <student name="David"></student>
  <student name="Craig"></student>
  <student name="Erik"></student>
</students>
```

```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="students">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="student" maxOccurs="unbounded" minOccurs="0">
          <xs:complexType>
            <xs:attribute type="xs:string" name="name" use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Exercise 4

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="person">
    <xs:complexType>
      <xs:attribute type="xs:string" name="name" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="food">
    <xs:complexType>
          <xs:attribute type="xs:string" name="name" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="eats">
    <xs:complexType>
          <xs:attribute type="xs:string" name="diner" use="optional"/>
          <xs:attribute type="xs:string" name="dish" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="meal">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="person" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element ref="food" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element ref="eats" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Is this XML matched?

```
<meal>
  <person name="Alice"/>
  <food name="salad"/>
  <eats diner="Alice" dish="salad"/>
  <person name="Bob"/>
  <food name="salad"/>
  <eats diner="Bob" dish="salad"/>
  <person name="Carol"/>
  <food name="sandwich"/>
  <eats diner="Carol" dish="sandwich"/>
</meal>
```

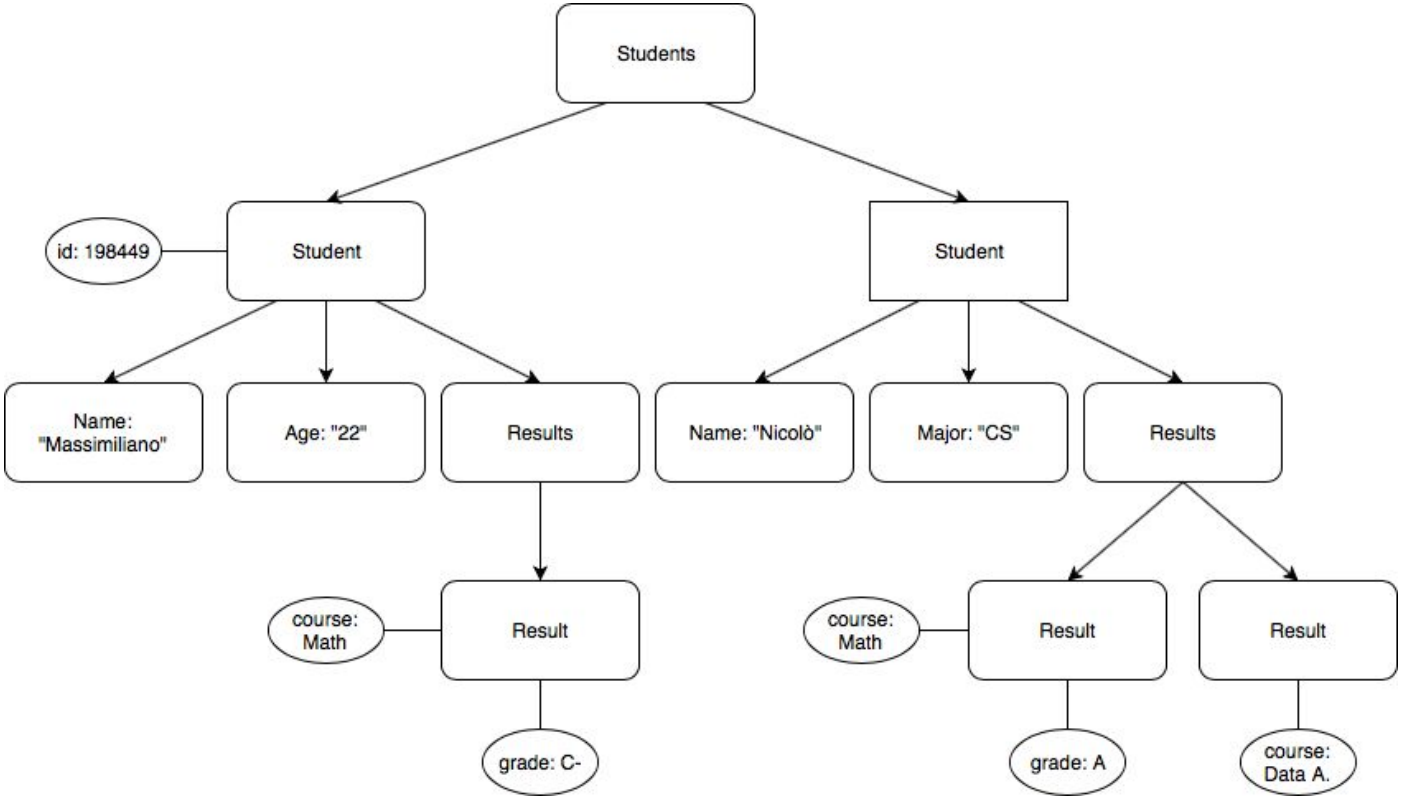
```
<meal>
  <person name="Alice"/>
  <person name="Bob"/>
  <person name="Carol"/>
  <person name="Dave"/>
  <food name="salad"/>
  <food name="turkey"/>
  <food name="sandwich"/>
  <eats diner="Alice" dish="turkey"/>
  <eats diner="Bob" dish="salad"/>
  <eats diner="turkey" dish="Dave"/>
</meal>
```

XPath

XPath

- Not mature as querying relational databases
- No underline algebra (implies also less intuitive query)
- XPath allows to look for specific paths in the underlying tree and to set up some simple conditions

XPath


Students
id: 198449
Student
Student
Name: "Massimiliano"
Age: "22"
Results
Name: "Nicolò"
Major: "CS"
Results
course: Math
Result
course: Math
Result
Result
grade: C-
grade: A
course: Data A.

XPath

Basic constraints:

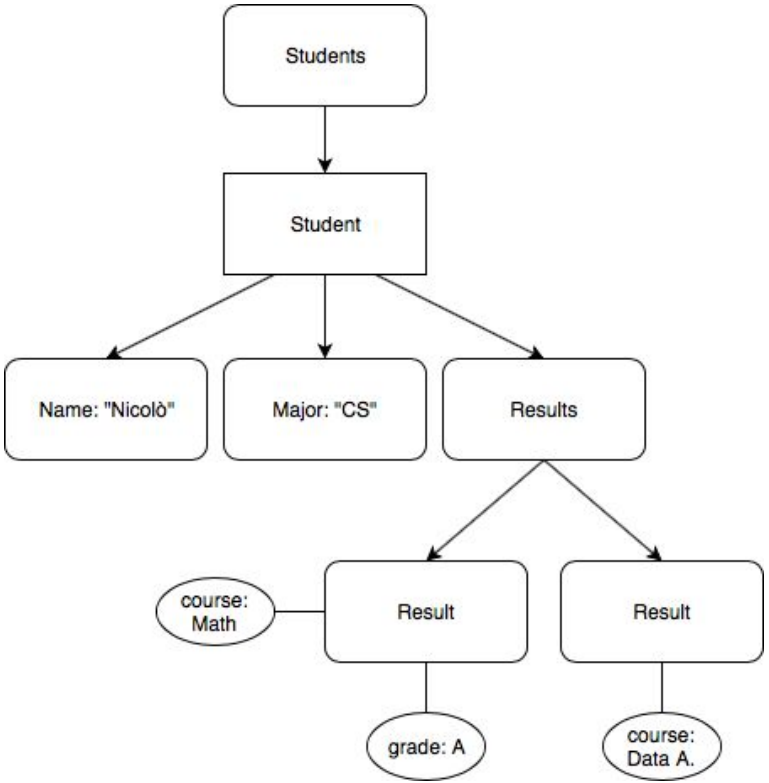
/ - to navigate the root element (it is also the separator)

<element_name> - to navigate through a specific element (as in SQL, * can be used)

@<attribute_name> - to access the value of an attribute

// - to match any descendant of the current node

[<cond>] - to evaluate a condition at a certain level



XPath

Access to “results”

//results

/students/student/results

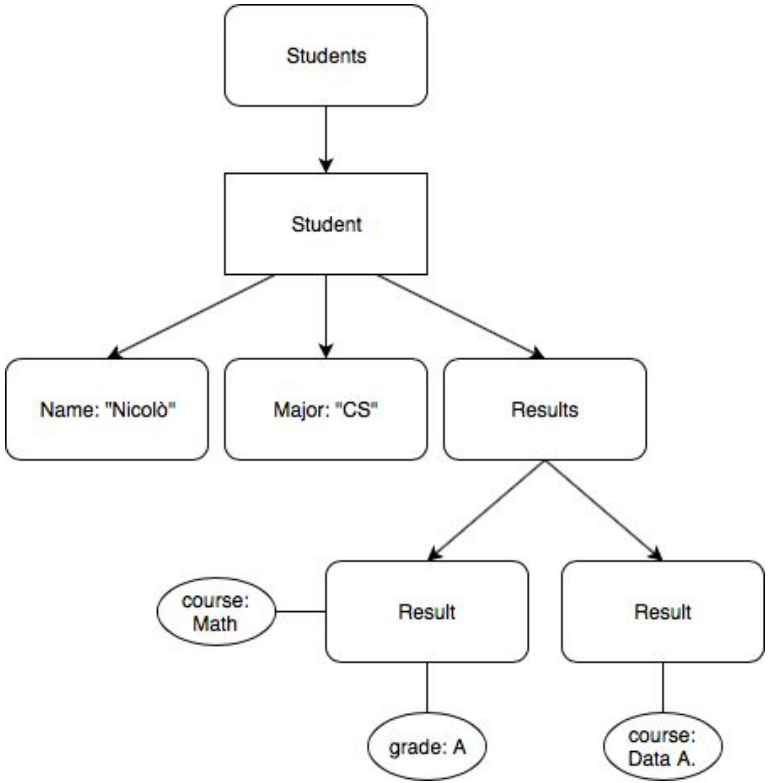
Access to a specific student

/students/student[1]

/students/student[last()]

Conditions

/students/student[name="Massimiliano"]/results



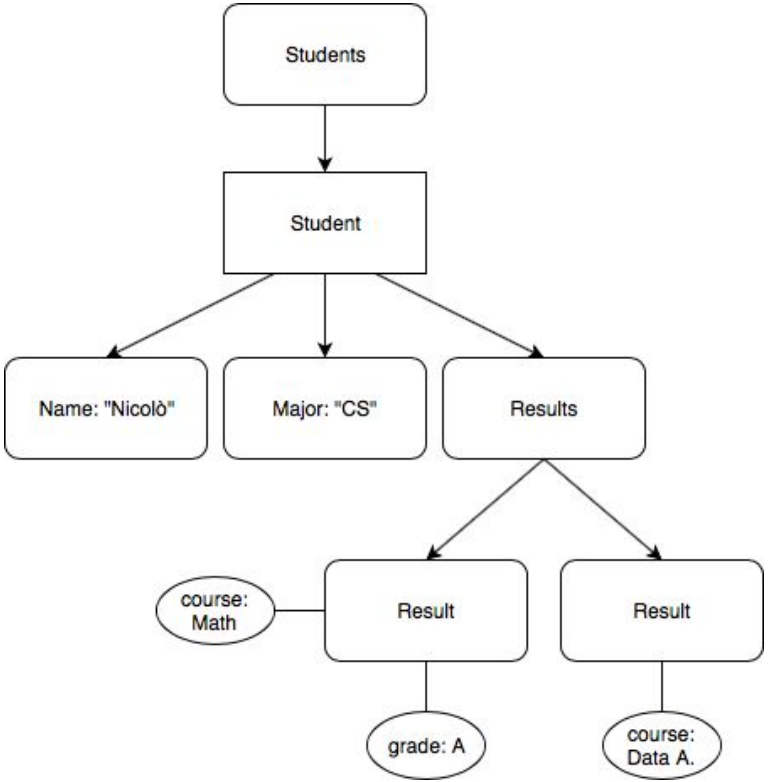
XPath

Access to all the results with the attribute grade

//result[@grade]

Filter on the attribute

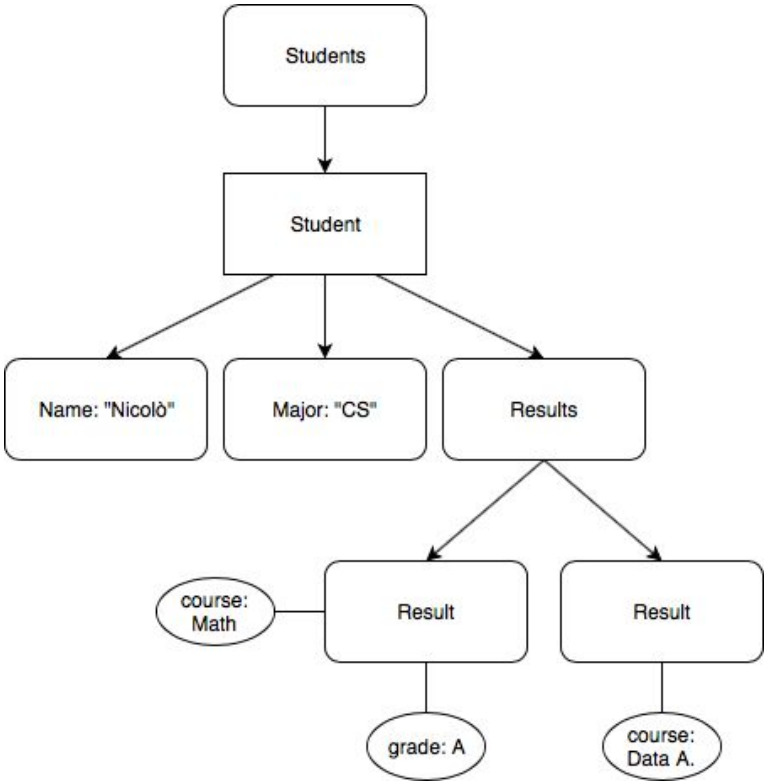
//result[@grade="A"]



XPath

Advanced constraints:

- built-in functions
  - contains(string1, string2)
  - name()
- Axes
  - parent::
  - descendants::
  - self::
  - … other 11



Practice

“Presentations aren't efficient without cats”
- Lex Fridman, research scientist @ MIT

Exercise 1

1. Return all the students older than 20
2. Return all the student that are studying computer science
3. Return all the students that took the “Data Analysis” class

```xml
<students>
  <student id="198449">
    <name>Massimiliano</name>
    <age>22</age>
    <major>Computer Science</major>
    <results>
      <result course="Math" grade="C-"/>
    </results>
  </student>
  <student id="192414">
    <name>Nicolo'</name>
    <major>Computer Science</major>
    <major>Secutiy and Network</major>
    <results>
      <result course="Math" grade="A"/>
      <result course="Data Analysis"/>
    </results>
  </student>
</students>
```

Exercise 1

1. //student[age>20] **or** /students/student[age>20]
2. //student[major="Computer Science"]/parent::*

Exercise 1

1. //student[age>20] **or** /students/student[age>20]
2. //student[major="Computer Science"]/parent::*
3. //student/results/result[@course="Data Analysis"]/ancestor::student

XQuery

XQuery

- Expression language (compositional)
- Work on sequences of elements
- Return sequences of elements
- FLOWR expression
    - For
    - Let
    - Order
    - Where
    - Return

XPath

Access to “results”

//results

/students/student/results

Access to a specific student

/students/student[1]

/students/student[last()]

Conditions

/students/student[name="Massimiliano"]/results

XQuery

Access to “results”

/students/student/results

Access to a specific student

for $x in /students/student

where $x/age>20

order by $x/age

return $x/major

Please, fill up this form

goo.gl/SkdN8V

Homework - D.L. 15 November 2018

Navigate to github.com/MassimilianoLuca/Introduction-to-XML/ and download “countries.xml” :

1. Generate the XSD file for the document given
2. Return the names of all countries where a city in that country contains more than one-third of the country's population (With XPath)
3. Return the average number of languages spoken in countries where Russian is spoken (With Xpath)
4. Return the names of all countries whose name textually contains a language spoken in that country. For instance, Uzbek is spoken in Uzbekistan, so return Uzbekistan (Optional, try to do it with XQuery)

Contact

Massimiliano Luca
massimiliano.luca@unitn.it
massimilianoluca.github.io

Nicolò Girardini
Nicolo .luca@unitn.it

Hope you enjoyed