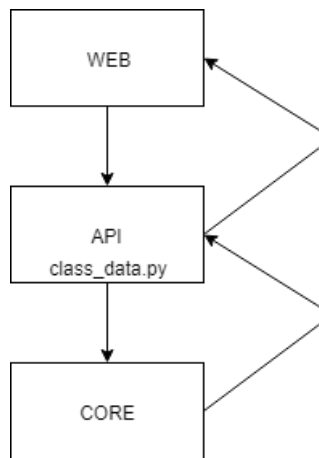


Create Class Plan

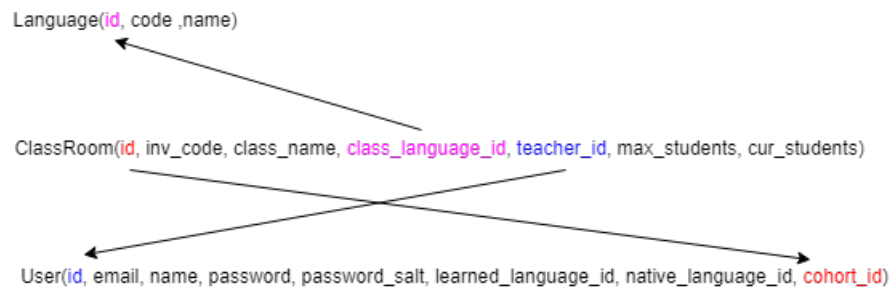
ZeeGuu dashboard team

March 2018

1 Overview



2 Relational mapping



3 Class Classroom

3.1 The following is pseudo-code for the Classroom class mimicking python

```
Class Classroom:
    id = db.Column(db.Integer , primary_key=True)
    inv_code = db.Column(db.String(255), unique = True)
    class_name = db.Column(db.String(255))
    class_language_id = db.Column(db.Integer , db.ForeignKey(Language.id))
    teacher_id = db.Column(db.Integer , db.ForeignKey(User.id))
    max_students = db.Column(db.Integer)
    cur_students = db.Column(db.Integer)

    class_language = relationship(Language, foreign_keys=[class_language_id])
    teacher = relationship(User, foreign_keys=[teacher_id])

    def __init__(inv_code , class_name , class_language_id , teacher_id , max_students):
        self.id = MaxClass()+1
        self.inv_code = inv_code
        self.class_name = class_name
        self.class_language_id = class_language_id
        self.teacher_id = teacher_id
        self.max_students = max_students
        self.cur_students = 0
```

3.2 Considerations

1. Instead of being given a '*inv_code*' the database could generate a new unique one itself. (Possibly from hashing of the Primary key)
2. Primary key increment is likely automated but I included it for the sake of clarity.
3. The Cohort class could also be adapted like this instead of the creation of the new Classroom class. (otherwise the class Cohort is no longer necessary)

4 Create user

4.1 The following is pseudo-code for the creation of Users using an invite code

```

(Take all other user inputs like name, email etc.)
Request inv_code
if (exists(inv_code) && allowed(inv_code))
    class_id = request_class_id(inv_code)
    new User(other stuff, class_id)
else
    output 'Invalid code!'
    return
end

```

5 Database functions

5.1 The following is pseudo-code for the database functions used in creation of user

(The SQL-python interaction may be inaccurate but this is just to model how it should perform)

```

exists(inv):
    int id =
    {
        SELECT id
        FROM ClassRooms
        WHERE inv_code = inv;
    };

    if(id != null)
        return True
    else
        return False
    end

allowed(inv):
    int max =
    {
        SELECT max_students
        FROM ClassRooms
        WHERE inv_code = inv;
    };
    int cur =
    {
        SELECT cur_students
        FROM ClassRooms
        WHERE inv_code = inv;
    };

    if(cur < max)

```

```

        return True
    else
        return False
    end

request_class_id(inv):
    int id =
    {
        SELECT id
        FROM ClassRooms
        WHERE inv_code = inv;
    };

    increase_students(id)
    return id

increase_students(id):
    int cur =
    {
        SELECT cur_students
        FROM ClassRooms
        WHERE id = id;
    };

    {
        UPDATE ClassRooms
        SET cur_students = cur+1
        WHERE id = id;
    }

```

5.2 More database functions relating to classes

```

get_students_from_class(class_id):
    int [] student_ids =
    {
        SELECT student_id
        FROM Users
        WHERE id = class_id
        ORDER BY student_id ASC;
    }
    return student_ids

get_students_from_teacher(teacher_id):
    int [] student_ids =

```

```

{
    SELECT student_id
    FROM Users
    JOIN ClassRooms ON Users.class_id = ClassRooms.id
    WHERE ClassRooms.teacher_id = teacher_id
    ORDER BY student_id ASC
    GROUP BY ClassRooms.id;

};
return students_ids

```

5.3 Other functions needed for Dashboard

- Get all classes belonging to a teacher.
- Get all info relevant to a class:
 - Get invite code to a class.
 - Get Max students from a class.
 - Get Current students from a class.
 - Get Name of class.
 - Get Language of class.