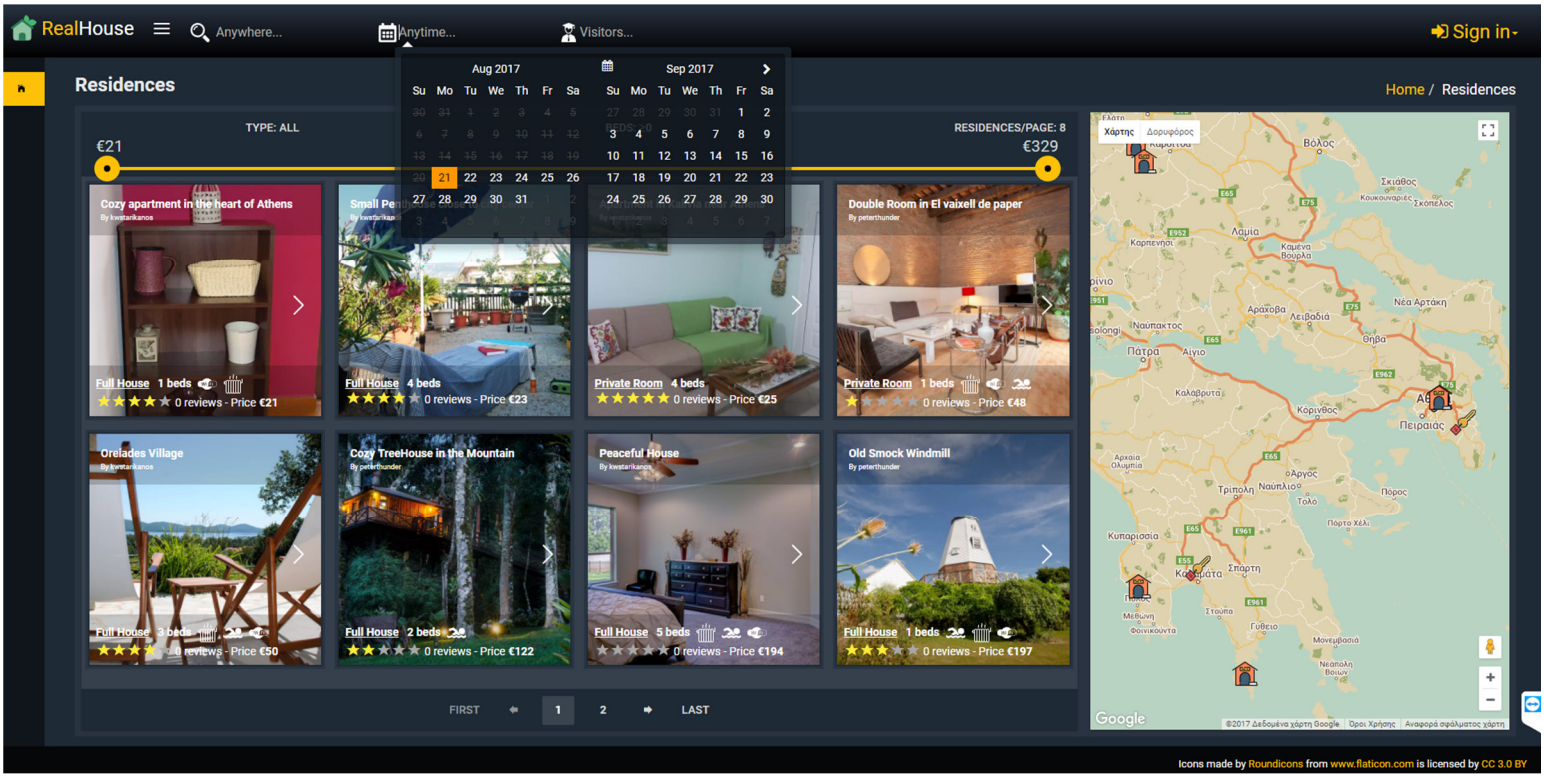
Τεχνολογιες εφαρμογων διαδικτυου

Προγραμματιστική εργασία

Εφαρμογή διαδικτύου ενοικίασης δωματίων/κατοικιών (τύπου Airbnb)



Ομάδα χρηστών 54

AM:1115201300106

Πέτρος Μορφίρης

AM:1115201300202

Κώστας Χατζόπουλος

Περιβάλλον υλοποίησης

Front-end/client

WebStorm JavaScript IDE

Back-end/server

IntelliJ IDEA

Χρήσιμα εργαλεία

POSTMAN

https://www.base64-image.de/

https://jsonformatter.curiousconcept.com/#

Λεπτομέρειες υλοποίησης FRONT-END/CLIENT

Η υλοποίηση της εφαρμογής έγινε με το framework angularjs της βιβλιοθήκης javascript και βασίστηκε στο template bluradmin (http://akveo.com/blur-admin/#/dashboard) όπου έγιναν σημαντικές αλλαγές ώστε να προσαρμοστεί στις ανάγκες της εφαρμογής.

για το στυλ-θέμα της, στο μεγαλύτερο μέρος του, έγινε χρήση της **angular material** σε συνδυασμό με **twitter bootstrap** καθώς μικρο-αλλαγές στα **css** των παραπάνω, κυρίως για τα χρώματα. επιπρόσθετα, έγινε χρήση της βιβλιοθήκης **google maps** για την εμφάνιση του χάρτη στα σημεία όπου ήταν απαραίτητο και χρησιμοποιήθηκε το **google maps location autocomplete api** για την αναζήτηση σημείων/πόλεων διαμονής, την εγγραφή νέων χρηστών και την εισαγωγή νέων δωματίων.

Double date picker:

Για την επιλογή ζεύγους ημερομηνιών (start-end) στην μπάρα αναζήτησης αλλά και στη σελίδα κάθε δωματίου έγινε χρήση του plugin angular-daterangepicker 0.2.2

Εικονίδια:

Τα εικονίδια βρέθηκαν στα font awesome icons και flaticon icons.

Slider:

Το slider εμφάνισης εικόνων είναι όλο custom από εμάς.

Pagination:

Το pagination (σελιδοποίηση στο πλέγμα) είναι μερικώς custom από εμάς.

Price slider:

Για την υλοποίηση του φίλτρου τιμών των δωματίων, χρησιμοποιήθηκε η βιβλιοθήκη angularjs-slider 6.3.0

Data tables:

Για την εμφάνιση πληροφοριών των χρηστών στο admin-panel και αντίστοιχα των δωματίων στο host-panel έγινε χρήση της βιβλιοθήκης datatables.net-dt 3.2.2, η οποία παρέχει και δυνατότητα εξαγωγής των πληροφοριών αυτών σε αρχείο μορφής pdf.

Pictures upload:

Για το ανέβασμα εικόνων κατά τη διάρκεια της εισαγωγής δωματίων χρησιμοποιήθηκε η βιβλιοθήκη lf-ng-md-file-input 1.5.2

Λεπτομέρειες υλοποίησης BACK-END/SERVER

Η διεπαφή προγραμματισμού της εφαρμογής (api) υλοποιήθηκε σε αρχιτεκτονική Rest (Representational State Transfer), η οποία, είναι μια αρχιτεκτονική client-server για τη μεταφορά αναπαραστάσεων πόρων (resources) με το πρωτόκολλο http. Ο κώδικας γράφηκε σε γλωσσά προγραμματισμού java και το framework της **Java** που χρησιμοποιήθηκε είναι το **Spring**.

Java persistence API (JPA):

Για τη διαχείρηση των οντοτήτων της βάσης (create, read, update, delete) χρησιμοποιήθηκε JPA. Ωστόσο, για την υλοποίηση της αναζήτησης δωματίων, μαζί με όλα τα φίλτρα, χρησιμοποιήθηκε JPA criteria api που παρέχει τη δυνατότητα υλοποίησης δυναμικών ερωτημάτων (queries).

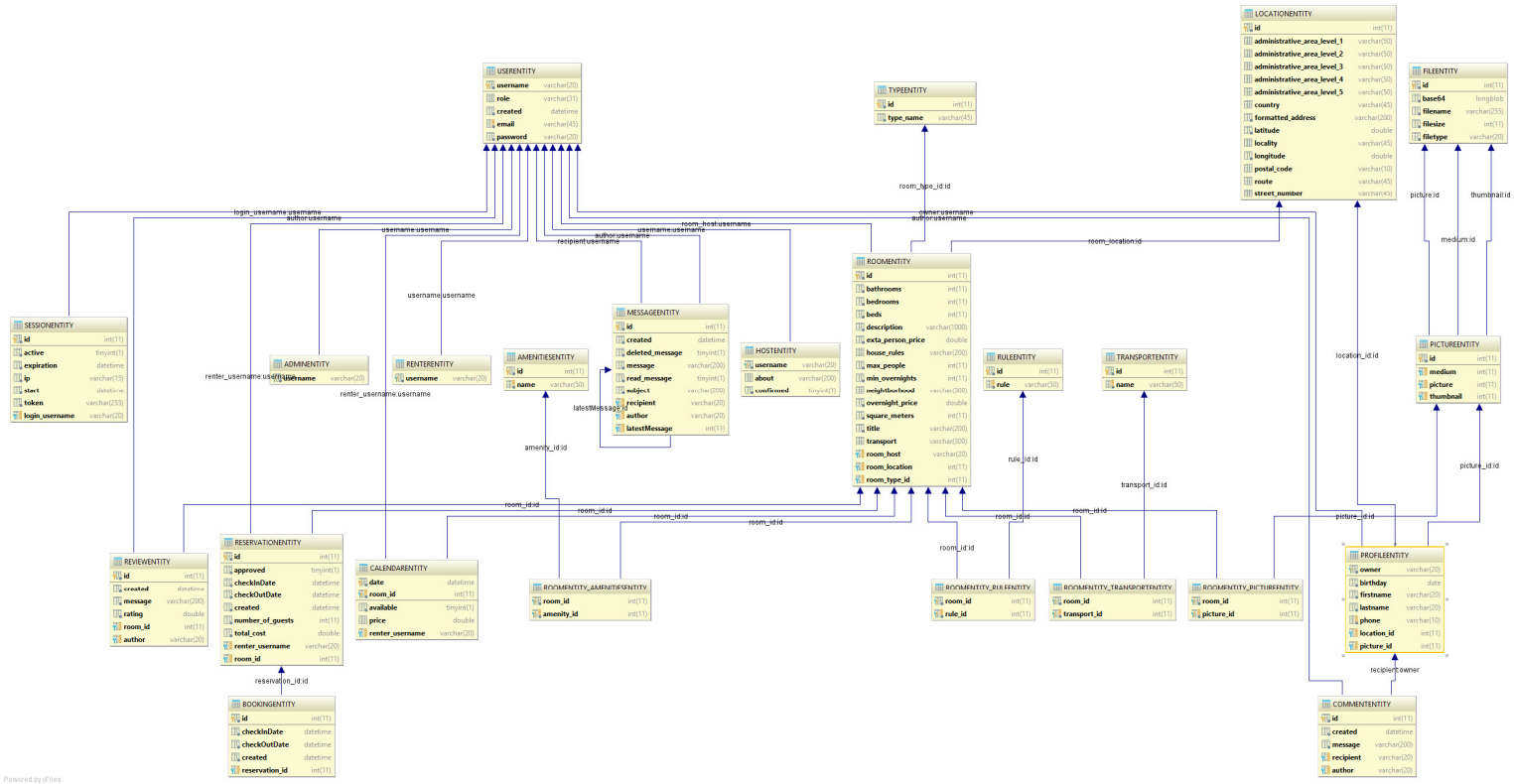
Json Web Token (jwt):

Έγινε χρήση της βιβλιοθήκης jwt 0.7.0 για την υλοποίηση του φίλτρου αυθεντικοποίησης (authentication filter).

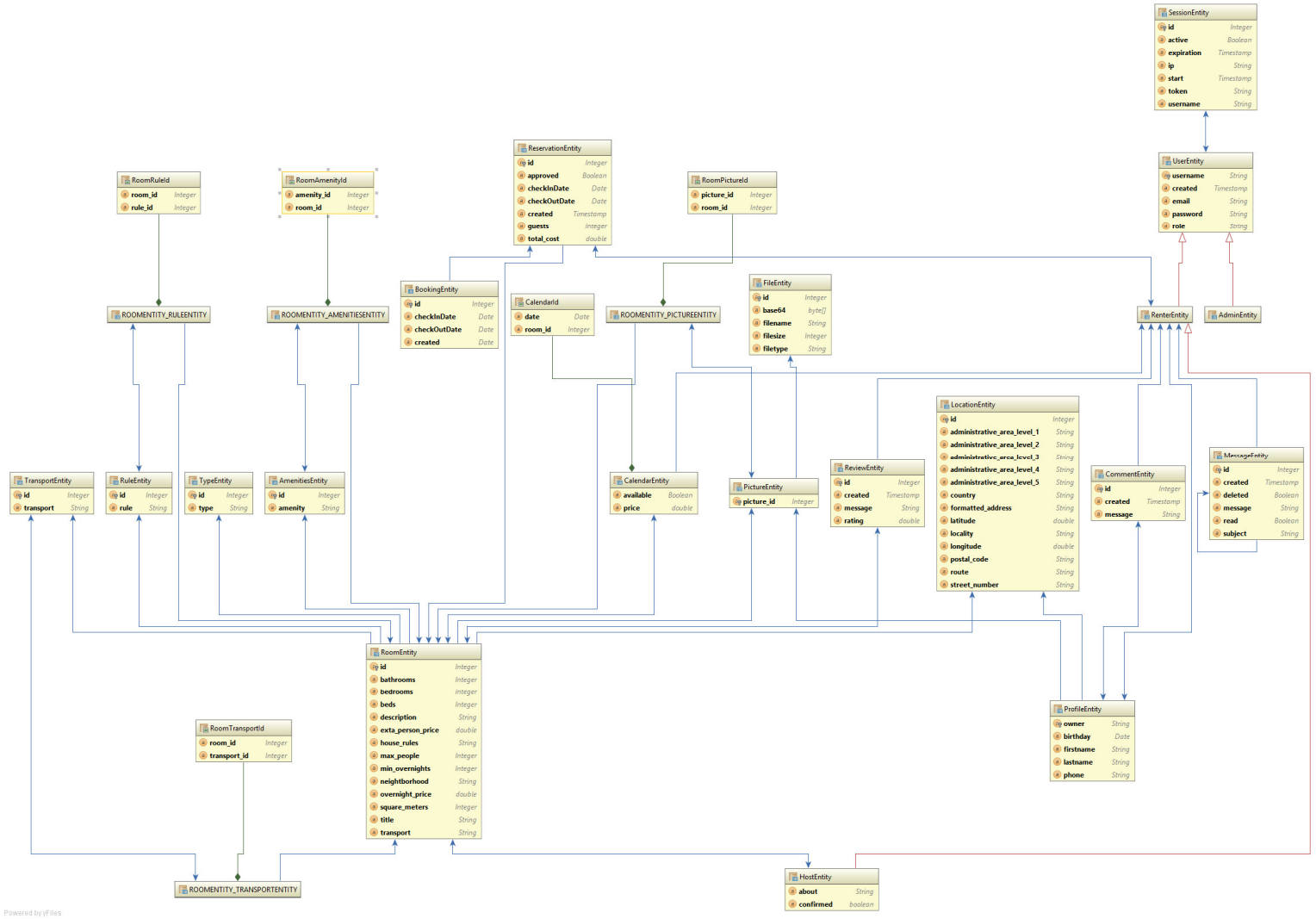
Validators:

για την επικύρωση των στοιχείων στις κλάσσεις-μοντέλα χρησιμοποιήθηκαν οι βιβλιοθήκες **javax.validation.constraints** και **org.hibernate.validator**

Αναπαράσταση του σχήματος οντοτήτων (jpa persistence unit)



Τελικό σχήμα της βάσης δεδομένων που δημιουργήθηκε από το persistence unit (jpa)



Web API Endpoint documentation

Roles:

admin, renter, host, owner

Headers:

content-type: application/json, στις μεθόδους post και put

authorization: required, όταν ζητούνται οι ρόλοι

User endpoint:

Users can be filtered by:

start, size, depth

METHOD	ENDPOINT	USAGE	RETURNS	ROLES
GET	/users?start={start}&size={size}&depth={depth}	Get all users	users	admin
GET	/users/{role}/show	Get users by role	users	admin
GET	/users/{username}	Get user	user	admin, owner
GET	/users/{username}/profile	Get user's profile	profile	-
GET	/users /{username}/profile/picture	Get profile's picture	file	-
GET	/users/ {username}/comments/	Get profile's comments	comments	-
GET	/users/{username}/messages/inbox	Get inbox messages	messages	owner
GET	/users/{username}/messages/outbox	Get outbox messages	messages	owner
GET	/users/{username}/messages/deletedMessages	Get deleted messages	messages	owner
GET	/users/{username}/messages/{id}	Get message	message	owner
POST	/users	Create user	user	admin
POST	/users/{username}/comments	Post comment	-	renter, host
POST	/users/{username}/messages/startConversation	Start a conversation	-	renter, host
POST	/users/{username}/message/answerMessage	Answer message	-	owner
PUT	/users/{username}	Edit user	-	admin, owner
PUT	/users /{username}/profile/picture	Edit profile's picture	-	owner
PUT	/users/{username}/comments/{comment_id}	Edit comment	-	owner
DELETE	/users/{username}	Delete user	-	admin, owner
DELETE	/users /{username}/profile/picture/{picture_id}	Delete profile's picture	-	owner
DELETE	/users/{username}/comments/{comment_id}	Delete comment	-	owner
DELETE	/users/{username}/message/{id}	Delete message	-	owner

Room endpoint

Rooms can be filtered by:

owner, start, size, type, beds, min_price, max_price, depth, locality, country, checkin, checkout, guests

METHOD	ENDPOINT	USAGE	RETURNS	ROLES
GET	/rooms?owner={owner}&start={start}…κλπ…	Get rooms filtered	residences	-
GET	/rooms/{id}	Get room	residence	-
GET	/rooms/mylistings/{username}/	Get my listings	rooms	owner
GET	/rooms/myreservations/{username}	Get my reservations	reservations	owner
GET	/rooms/{id}/picture/{picture_id}	Get room's picture by id	file	-
GET	/rooms/{id}/reviews/	Get room reviews	reviews	-
POST	/rooms	Insert a room	-	host
POST	/rooms/{id}/setCalendarEntries	Set room calendar entries	-	owner
POST	/rooms/{id}/picture	Insert room pictures	-	owner
POST	/rooms/{id}/reviews	Post review	-	renter, host
PUT	/rooms/{id}	Update room	-	owner
PUT	/rooms /{id}/approveReservation/{reservation_id}	Approve reservation	-	owner
PUT	/rooms/{id}/reviews/{review_id}	Edit review	-	admin, owner
DELETE	/rooms/{id}/deleteCalendarEntries	Delete room calendar entries	-	owner
DELETE	/rooms/{id}	Delete room	-	owner
DELETE	/rooms/{id}/picture/{picture_id}	Delete room picture	-	owner
DELETE	/rooms/cancelReservation/{reservation_id}	Cancel reservation	-	owner
DELETE	/rooms/{id}/reviews/{review_id}	Delete review	-	admin, owner

Authentication endpoint:

METHOD	ENDPOINT	USAGE	RETURNS	ROLES
GET	/authentication/logout	Check valid token users	-	Admin, renter, host, owner
GET	/authentication/check/username/{username}	Check if username exists	UsernameObj	-
GET	/authentication/check	Check valid token and logged in users	-	Admin, renter, host, owner
GET	/authentication/check/email/{email}	Check if the email already exists	EmailObj	-
GET	/ authentication/ check/phone/{phone}	Check if the phone already exists	PhoneObj	-
POST	/ authentication/login	User login	AccessObj	-
POST	/ authentication/register	User register	-	-

File endpoint:

METHOD	ENDPOINT	USAGE	RETURNS	ROLES
GET	/files/picture/{picture_id}	Get picture	file	-