ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΜΕ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΜΙΚΡΟΥΠΗΡΕΣΙΩΝ

Νικόλαος Μπάγιας

Μεταπτυχιακό Εφαρμοσμένης Πληροφορικής

Επιβλέπων Καθηγητής: Κωνσταντίνος Μαργαρίτης

Ιούνιος 2020

Μικρουπηρεσίες

Μονολιθική Αρχιτεκτονική

Service-oriented Αρχιτεκτονική

Η αρχιτεκτονική των μικρουπηρεσιών δίνει έμφαση στην αποσύνθεση μιας εφαρμογής σε μικρότερες και ανεξάρτητες υπηρεσίες. Ο στόχος τους είναι να παραμείνει το σύστημα ευέλικτο και προσαρμόσιμο, καθώς η πολυπλοκότητα του αυξάνεται.

Η μονολιθική αρχιτεκτονική στρέφεται προς την αντίθετη κατεύθυνση δομώντας μια εφαρμογή ως ένα ολοκληρωμένο εκτελέσιμο. Ότι χάνει σε ευελιξία το κερδίζει σε απλότητα.

Η service-oriented αρχιτεκτονική θεωρείται ο πρόδρομος των μικρουπηρεσιών, καθώς το σύστημα αποτελείται και εδώ από αρθωτές υπηρεσίες. Η διαφορά με τις μικρουπηρεσίες έγκειται περισσότερο στο διαφορετικό πεδίο εφαρμογής που επικεντρώνεται η διαδικασία αποσύνθεσης. Η SOA προσπαθεί να εξυπηρετήσει όλο το επιχειρηματικό πεδίο ταυτόχρονα, ενώ οι μικρουπηρεσίες συγκεκριμένη λειτουργικότητα.

Πλεονεκτήματα

Τεχνολογική ετερογένεια

Κάθε μικρουπηρεσία έχει τη δική της τεχνολογική στοίβα και το δικό της μοντέλο δεδομένων, δίνοντας τη δυνατότητα επιλογής των κατάλληλων εργαλείων.

Ανθεκτικότητα

Η διάσπαση του συστήματος σε μικρότερα μέρη προσφέρει προστασία από μετάδοση σφαλμάτων στο υπόλοιπο σύστημα.

Κλιμάκωση

Κάθε μικρουπηρεσία έχει την δική της διεργασία επιτρέποντας την επιλεκτικη κλιμάκωση του συστήματος.

Ευκολία δημοσίευσης (deployment)

Η κυκλοφορία μιας νέας έκδοσης μπορεί να περιοριστεί σε μία μόνο μικρουπηρεσία.

Προκλήσεις

Διαχείριση και Παρακολούθηση συστήματος

Η ύπαρξη πολλών συνεργαζόμενων υπηρεσιών αυξάνει τις απαιτήσεις για διαφάνεια του συστήματος.

Εξαρτήσεις κώδικα

Η ευκολία της δημοσίευσης είναι κάτι που θέλει ενεργή προσπάθεια, καθώς οι εξαρτήσεις σε κοινές βιβλιοθήκες προσθέτουν ανάγκη για συντονισμό.

Ενσωμάτωση Γραφικής Διεπαφής

Οι μικρουπηρεσίες επικεντρώνεται στην διαίρεση του λειτουργικού μέρους του συστήματος, δημιουργώντας δυνητικά δυσκολίες στην διαχείριση της γραφικής διεπαφής.

Μοντελοποίηση μικρουπηρεσιών

Διάχυτη ορολογία (Ubiquitous Language)

Μία βασική έννοια του Domain-Driven-Design είναι η καθιέρωση της διάχυτης ορολογίας. Όλο το λεξιλόγιο και η γνωσιακή βάση που χρησιμοποιούν οι ειδικοί του οργανισμού πρέπει συστηματικά να καταγραφεί και να αποτελεί κομμάτι της τεχνικής υλοποίησης. Στόχος είναι ότι μέσα από αυτό θα βελτιωθεί η επικοινωνία τεχνικών και ειδικών αλλά και ότι η αρχιτεκτονική του συστήματος θα αντικατοπτρίζει την επιχειρηματική πραγματικότητα.

Οριοθετημένο πλαίσιο (Bounded Context)

Το Domain-Driven-Design προτρέπει την διαίρεση των μοντέλων ενός οργανισμού σε οριοθετημένα πλαίσια. Αρχικά το μοντέλο πρέπει να είναι συνεπές και να μην περιέχει αντικρουόμενες έννοιες. Πολλές φορές μία οντότητα θα εμφανίζεται σε διαφορετικά πλαίσια αλλά εφόσον το μοντέλο είναι σωστό δεν αποτελεί πρόβλημα αλλά επιθυμητό χαρακτηριστικό. Οι μικρουπηρεσίες προσπαθούν να επωφεληθούν από αυτό το διαχωρισμό δημιουργώντας τον ευριστικό κανόνα πως μια μικρουπηρεσία πρέπει να είναι ένα οριοθετημένο πλαίσιο.

*Domain-Driven-Design: Μια προσέγγιση σχεδιασμού συστημάτων που δίνει έμφαση στο domain, με εξαγωγή μοντέλων και διαδικασιών από αυτό.

Τεχνολογίες Ενσωμάτωσης και Επικοινωνίας

REST (Representational State Transfer)

Το REST χρησιμοποιεί το πρωτόκολλο HTTP για να προσφέρει διαχείριση διαδικτυακών πόρων. Με την χρήση του μοντέλου διακομιστή πελάτη, όπου ο πελάτης ζητάει έναν πόρο και περιμένει απάντηση εξυπηρετεί περισσότερο μια σύγχρονη μορφή επικοινωνίας.

AMQP (Asynchronous Message Queue Protocol)

Το AMQP προσφέρει έναν τρόπο ασύγχρονης επικοινωνίας με την χρήση ουρών μηνυμάτων. Με την διαμεσολάβηση ενός διακομιστή τα μηνυματα δρομολογούνται στις κατάλληλες ουρές. Οι μικρουπηρεσίες εξυπηρετούνται ιδιαίτερα καλά από αυτό το μοτίβο επικοινωνίας.

gRPC

Το gRPC προσφέρει την τεχνική RPC με χρήση Protocol Buffers ως πρωτόκολλο μεταφοράς. Η αξιοποίηση του HTTP/2 δίνει δυνατότητες αμφίδρομης μετάδοσης δεδομένων, κάνοντας το καλό υποψήφιο για επικοινωνία μικρουπηρεσιών.

WebSockets

Το WebSockets είναι ένα πρωτόκολλο μεταφοράς δεδομένων μέσω TCP. Συνήθως χρησιμοποιείται για την ασύγχρονη μετάδοση δεδομένων από έναν διακομιστή στον φυλλομετρητή ενός συνδεδεμένου χρήστη.

Εφαρμογή BookSpot

Καταγραφή απαιτήσεων

Για την επίδειξη της αρχιτεκτονικής των μικρουπηρεσιών αναπτύχθηκε μία εφαρμογή βαθμολόγησης βιβλίων. Οι απαιτήσεις από την εφαρμογή είναι σχετικά απλές αλλά αρκετά εκφραστικές ώστε να δικαιολογούν ένα ευρύ τεχνολογικό φάσμα.

Οι βασικές απαιτήσεις είναι:

- Σύνδεση και εγγραφή χρήστη
- Παρουσίαση και βαθμολόγηση βιβλίων
- Ταξινόμηση/φιλτράρισμα/αναζήτηση βιβλίων
- Εμφάνιση ανακοινώσεων σε συνδεδεμένους χρήστες

Αρχιτεκτονική συστήματος

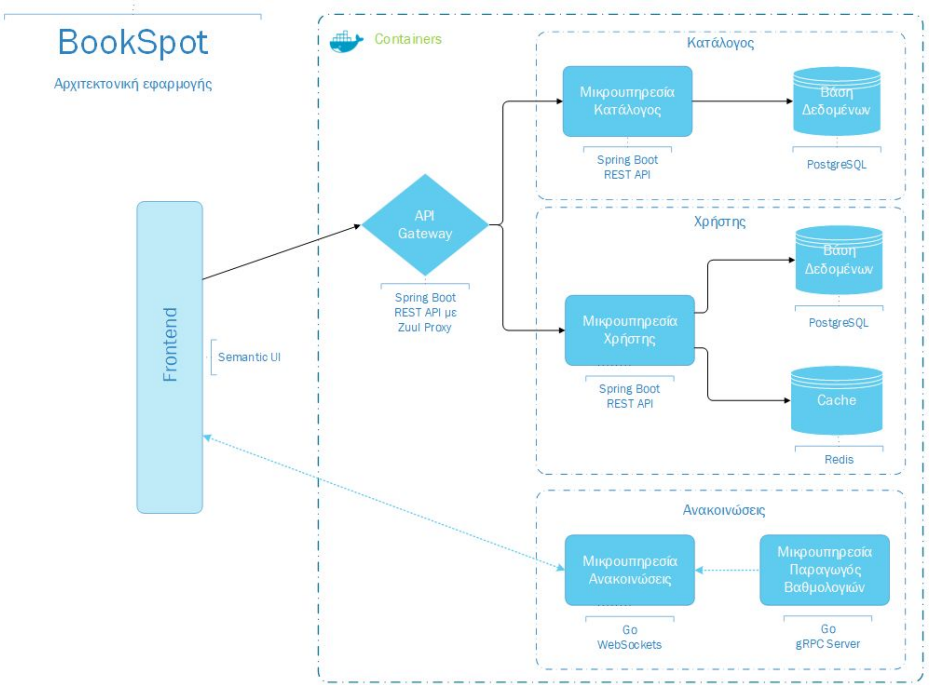
Θα αποσυνθέσουμε το τομέα της εφαρμογής σε τρεις λειτουργικούς υποτομείς, δημιουργώντας τέσσερα οριοθετημένα πλαίσια και ένα υποστηρικτικό.

Κατάλογος: Μια μικρουπηρεσία που προσφέρει πρόσβαση στις εγγραφές των βιβλίων (CRUD).

Χρήστης: Μια μικρουπηρεσία που παρέχει λειτουργίες σύνδεσης και εγγραφής για τους χρήστες.

Ανακοινώσεις: Εμφάνιση ειδοποιήσεων με χρήση δύο μικρουπηρεσιών, η μία ως παραγωγός και η άλλη ως ενδιάμεσος για την προώθηση τους στον χρήστη.

Πύλη API: Ξεχωριστή μικρουπηρεσία που δέχεται όλα τα αιτήματα και τα δρομολογεί στην κατάλληλη μικρουπηρεσία.



Εργαλεία ανάπτυξης

Docker: Εργαλείο εικονικοποίησης σε επίπεδο λογισμικού που συμβάλει στην αναπαραγωγιμότητα του περιβάλλοντος του εκτελέσιμου.

Docker-compose, multi-stage build, alpine

PostgreSQL: Σχεσιακό σύστημα βάσεων δεδομένων, ανοιχτού κώδικα, που προσφέρει αρκετά ευρύ φάσμα λειτουργικότητας.

One database per microservice, SQL dump

Spring Boot: Εργαλειοθήκη ανάπτυξης Java εφαρμογών με έμφαση στην ευκολία ανάπτυξης. Οι στόχοι του συμπίπτουν με τις ανάγκες των μικρουπηρεσιών.

Spring JPA/Spring REST/Spring Web

Redis: In-memory βάση δεδομένων για εγγραφές τύπου κλειδιού-τιμής.

Cache, TTL (Time-to-live)

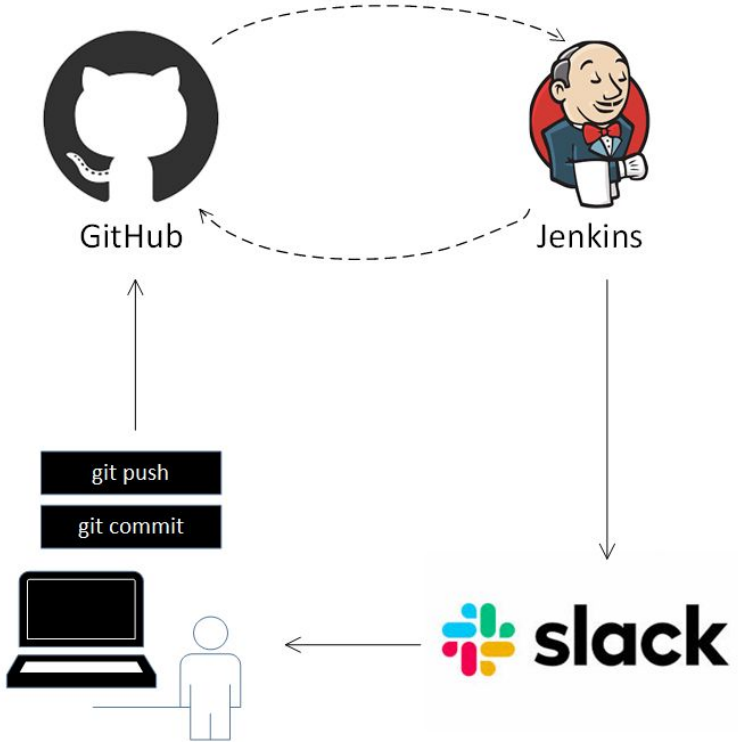
Semantic UI: JQuery εργαλειοθήκη για την ανάπτυξη της γραφικής διεπιφάνειας.

HTML, CSS, JavaScript

Διαδικασία ανάπτυξης

Η διαδικασία ανάπτυξης λογισμικού υποστηρίχθηκε από μια σειρά εργαλείων. Όλες οι προσθήκες και αλλαγές στον πηγαίο κώδικα γίνανε με το σύστημα ελέγχου εκδόσεων Git. Κάθε μία προώθηση των αλλαγών στον ιστότοπο GitHub (ένα αποθετήριο Git) πυροδοτεί την εκτέλεση δύο εργασιών, την μεταγλώττιση του κώδικα και, μετά, την εκτέλεση των τεστ. Το αποτέλεσμα αυτής της διαδικασίας προωθείται στο εργαλείο Slack, μια πλατφόρμα ανταλλαγής μηνυμάτων.

Πιο αφαιρετικά, κάθε ένα επιθυμητό χαρακτηριστικό διαιρούνταν σε εργασίες(tasks) με το σύστημα διαχείρισης πρότζεκτ Jira. Κάθε ένα task ήταν ένα διαφορετικό branch και περνούσε την προαναφερθείσα διαδικασία ξεχωριστά, επιτρέποντας την παράλληλη ανάπτυξη της εφαρμογής.



Μικρουπηρεσία Κατάλογος

Η μικρουπηρεσία Κατάλογος δίνει πρόσβαση στα διαθέσιμα βιβλία με όλες τους τις πληροφορίες καθώς και δυνατότητες πλοήγησης σε συναφείς πόρους (χρησιμοποιείται το πρότυπο HAL). Με τη χρήση των βιβλιοθηκών Spring REST/JPA, η δημιουργία της παραπάνω λειτουργικότητας μπορεί να επιτευχθεί απλά και μόνο επεκτείνοντας το κατάλληλο αποθετήριο.

```
public interface BookRepository extends PagingAndSortingRepository<Book, Long> {

   @RestResource(path = "/byRatingAsc", rel = "/byRatingAsc")
   Page<Book> findAllByOrderByRatingAsc(Pageable pageable);

   @RestResource(path = "/byTitle", rel = "/byTitle")
   List<Book> findBooksByTitleContainingIgnoreCase(@Param("title") String title);
}
```

http://localhost:8090/catalog/books/search/byTitle?title=philosophy


```
{
  "_embedded" : {
    "books" : [ {
      "title" : "Breakfast with Socrates: the philosophy of everyday life",
      "description" : "**A very ordinary day in the company of some...",
      "rating" : 1.0,
      "isbn" : "1afbb9d9-bdcf-4393-b4bb-7d391a4f926d",
      "language" : null,
      "edition" : 0,
      "cover_id" : 6349615,
      "publication_year" : 2010,
      "page_count" : 192,
      "_embedded" : {
...
      "_links" : {
    "self" : {
      "href" :
"http://localhost:8090/catalog/books/search/byTitle?title=philosophy"
    }
  }
}
```

Dockerfile

```
# Using global ARG in order to access main class on the second build stage.
# More info: https://github.com/moby/moby/issues/37345
ARG MAIN_CLASS

FROM gradle:5.2.1-jdk8-alpine AS build

LABEL maintainer="mai18043@uom.edu.gr"
# Gradle needs permission to build
COPY --chown=gradle:gradle . /workspace/src

WORKDIR /workspace/src

RUN gradle clean build --no-daemon
# Inflate application jar to take advantage of layered dependencies
RUN mkdir -p build/dependency && \
   cd build/dependency && \
   jar -xf ../libs/*.jar
# Use JRE version on runtime
FROM openjdk:8-jre-alpine

ARG MAIN_CLASS
ENV ENV_MAIN_CLASS=${MAIN_CLASS}

ARG SRC_PATH=/workspace/src/build/dependency
COPY --from=build ${SRC_PATH}/BOOT-INF/lib /app/lib
COPY --from=build ${SRC_PATH}/META-INF /app/META-INF
COPY --from=build ${SRC_PATH}/BOOT-INF/classes /app

ENTRYPOINT java -cp "app:app/lib/*"
com.github.bagiasn.bookspot.${ENV_MAIN_CLASS}
```

Μικρουπηρεσία Χρήστης

Η μικρουπηρεσία Χρήστης εξυπηρετεί κυρίως τις λειτουργίες της σύνδεσης και εγγραφής. Για την ολοκλήρωση αυτής της διαδικασίας απαιτείται τόσο η εκτέλεση ενός αιτήματος προς την βάση δεδομένων για την προσκόμιση του κωδικού όσο και προς το Redis για την αποθήκευση ενός αναγνωριστικού. Η αποφυγή διπλότυπων εγγραφών αφήνεται στην Postgres με έναν περιορισμό μοναδικότητας στο πεδίο της ηλεκτρονικής διεύθυνσης.

```
@RequestMapping(value = "/login", method = RequestMethod.POST)
public ResponseEntity<?> login(@RequestBody Credentials credentials) {
    String providedEmail = credentials.getEmail();
    logger.info("Login request received: {}", providedEmail);

    String userToken = redisTemplate.opsForValue().get(providedEmail);
    if (userToken != null && !userToken.isEmpty()) {
        logger.info("User already logged-in");
        return ResponseEntity.ok().body("User is already logged-in");
    } else {
        // First, find the requested user.
        User user = userRepository.findByEmail(providedEmail);
        if (user == null) {
            logger.warn("Could not find user with email {}", providedEmail);
            return ResponseEntity.notFound().build();
        } else {
            // Check if the password is correct.
            String providedPassword = HashGenerator.GetPasswordHash(credentials.getPassword());
            if (providedPassword.equals(user.getPassword())) {
                logger.info("Password is correct.");
                // Generate a token
                String token = UUID.randomUUID().toString();
                // Update redis.
                redisTemplate.opsForValue().set(providedEmail, token, Duration.ofSeconds(UserTokenTtl));
                Credentials creds = new Credentials();
                creds.setEmail(providedEmail);
                creds.setToken(token);

                return ResponseEntity.ok().body(creds);
            } else {
                logger.info("Wrong password was provided.");

                return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();
            }
```

Μικρουπηρεσία Gateway API

Με την χρήση της βιβλιοθήκης Zuul, όλα τα αιτήματα δρομολογούνται στις μικρουπηρεσίες Κατάλογος και Χρήστης αντίστοιχα. Πέρα από το ποια routes εξυπηρετεί πρέπει να θέσουμε και την τοποθεσία της μικρουπηρεσίας. Θα το κάνουμε μέσα από το docker-compose.

```
spring.application.name=gateway

zuul.routes.catalog.path=/api/catalog/**
zuul.routes.catalog.service-id=catalog

zuul.routes.user.path=/api/user/**
zuul.routes.user.service-id=user

ribbon.eureka.enabled=false

server.port=8080
```

Αρχείο ρυθμίσεων

```
gateway-api:
  image: bookspot/gateway-api
  container_name: gateway
  build:
    context: ./gateway
    args:
      - MAIN_CLASS=gateway.GatewayApplication
  environment:
    - ZUUL_ROUTES_CATALOG_URL=http://catalog-api:8090/catalog
    - ZUUL_ROUTES_USER_URL=http://user-api:8100/user
  ports:
    - "8080:8080"
  depends_on:
    - catalog-api
    - user-api
```

docker-compose

Μικρουπηρεσία Παραγωγός Βαθμολογιών

Για την προσομοίωση δραστηριότητας ενός χρήστη δημιουργήθηκε μια μικρουπηρεσία που παράγει τυχαίες βαθμολογήσεις βιβλίων. Η λειτουργικότητα προσφέρεται μέσα από έναν διακομιστή gRPC, που για κάθε ένα πρόγραμμα-πελάτη που συνδεέται ξεκινα μια διαδικασία όπου παράγει ένα τυχαίο όνομα και μια βαθμολογία και σε συνδυασμό με τη τυχαία επιλογή ενός τίτλου στέλνει ανα ένα χρονικό διάστημα το αποτέλεσμα (server-side-streaming).

```
func (s *announcementServer) Listen(in *pb.Request, stream
pb.Announcement_ListenServer) error {
  log.WithFields(log.Fields{
     "clientId": in.ClientId,
  }).Info("Received client request")

  data := fakeAnnouncement{}

  for {
     // Populate announcement with fake data.
     _ = faker.FakeData(&data)
     // Get a random int as book index.
     index := rand.Intn(len(bookData.Names))
     message := fmt.Sprintf("%v rated \"%v\" with %d", data.Username,
bookData.Names[index], data.Rating)
     // Send to client.
     if err := stream.Send(&pb.AnnouncementMessage{Message: message}); err != nil {
        return err
     }

     time.Sleep(time.Duration(data.Delay) * time.Second)
  }
}
```

Μικρουπηρεσία Ανακοινώσεις

Η τελευταία μικρουπηρεσία έχει αναλάβει την προώθηση των ειδοποιήσεων στον χρήστη. Διατηρεί ταυτόχρονα μια σύνδεση με την μικρουπηρεσία Παραγωγός Βαθμολογιών για να ακούει τα παραγόμενα μηνύματα αλλά και έναν διακομιστή WebSockets που επιτρέπει σε συνδεδεμένους χρήστες να λαμβάνουν αυτές τις ανακοινώσεις.

```
func (m *MessageBus) receiveAnnouncements(client pb.AnnouncementClient) {
  log.Info("Starting announcement client.")

  stream, err := client.Listen(context.Background(), &pb.Request{ClientId:
int32(23)})
  if err != nil {
     log.Fatalf("%v.Listen(_) = _, %v", client, err)
  }

  for {
     announcement, err := stream.Recv()
     if err == io.EOF {
        break
     }
     if err != nil {
        log.Fatalf("%v.Recv(_) = _, %v", client, err)
     }

     log.Debug(announcement.Message)

     m.messages <- announcement.Message
  }
}
```

BookSpot

docker-compose -f docker-compose.all.yml up

Το τελικό αποτέλεσμα φαίνεται στην εικόνα.

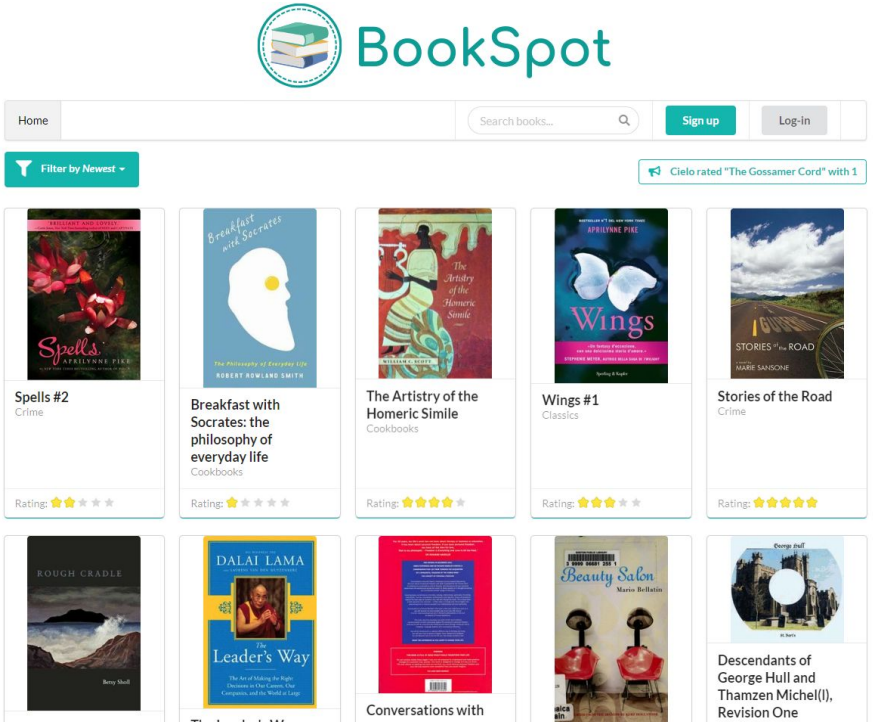
Χρήσιμοι σύνδεσμοι:

GitHub: https://github.com/bagiasn/book-spot

Jenkins: https://jenkins.book-spot.club/

JIRA: https://nbagias.atlassian.net/

Εφαρμογή: https://www.bookspot.dev/



Ευχαριστώ για την παρακολούθηση!

Νικόλαος Μπάγιας
mai18043@uom.edu.gr