Sigurnost i DevOps

Exception Handling

Logging & Auditing

Monitoring

ОБАВЕШТЕЊЕ ЗА СТУДЕНТЕ

- Настава на предмету Развој безбедног софтвера подразумева изучавање различитих механизама којима се нарушава информациона безбедност и врше напади на интернет апликације и софтверске системе.
- Студенти на предмету Развој безбедног софтвера могу ове методе за потребе изучавања да користе искључиво у оквиру затвореног лабораторијског окружења које је обезбеђено за наставу на предмету Развој безбедног софтвера.
- Студенти не могу да подразумевају да су на било који начин охрабрени од стране наставника или да им се препоручује да користе ове методе који се изучавају према другим апликацијама Електротехничког факултета или апликацијама било ког трећег правног или физичког лица.
- Свака евентуална активност коју би предузео неки студент коришћењем ових метода и механизама према апликацијама које нису у оквиру лабораторије на предмету искључива је одговорност студента.

DevOps

- Set praksi koje kombinuju razvoj (Dev) i operaciju (Ops) softvera.

- Cilj je skraćivanje ciklusa razvoja čestim dostavljanjem novih verzija i povećanje kvaliteta softvera u razvoju i operaciji.

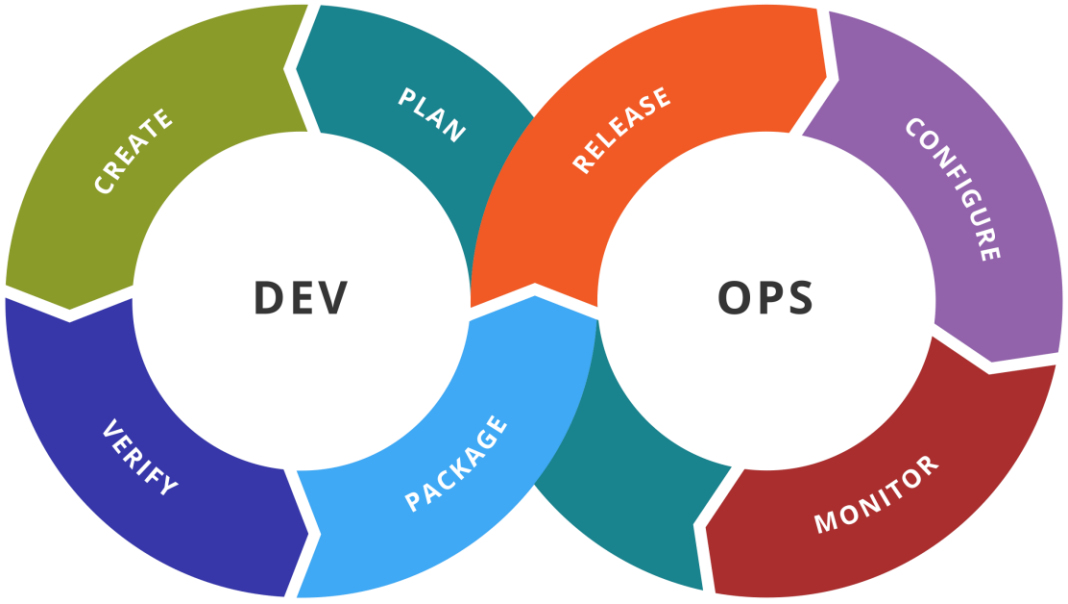
DevOps – Development & Operations

Uključuje

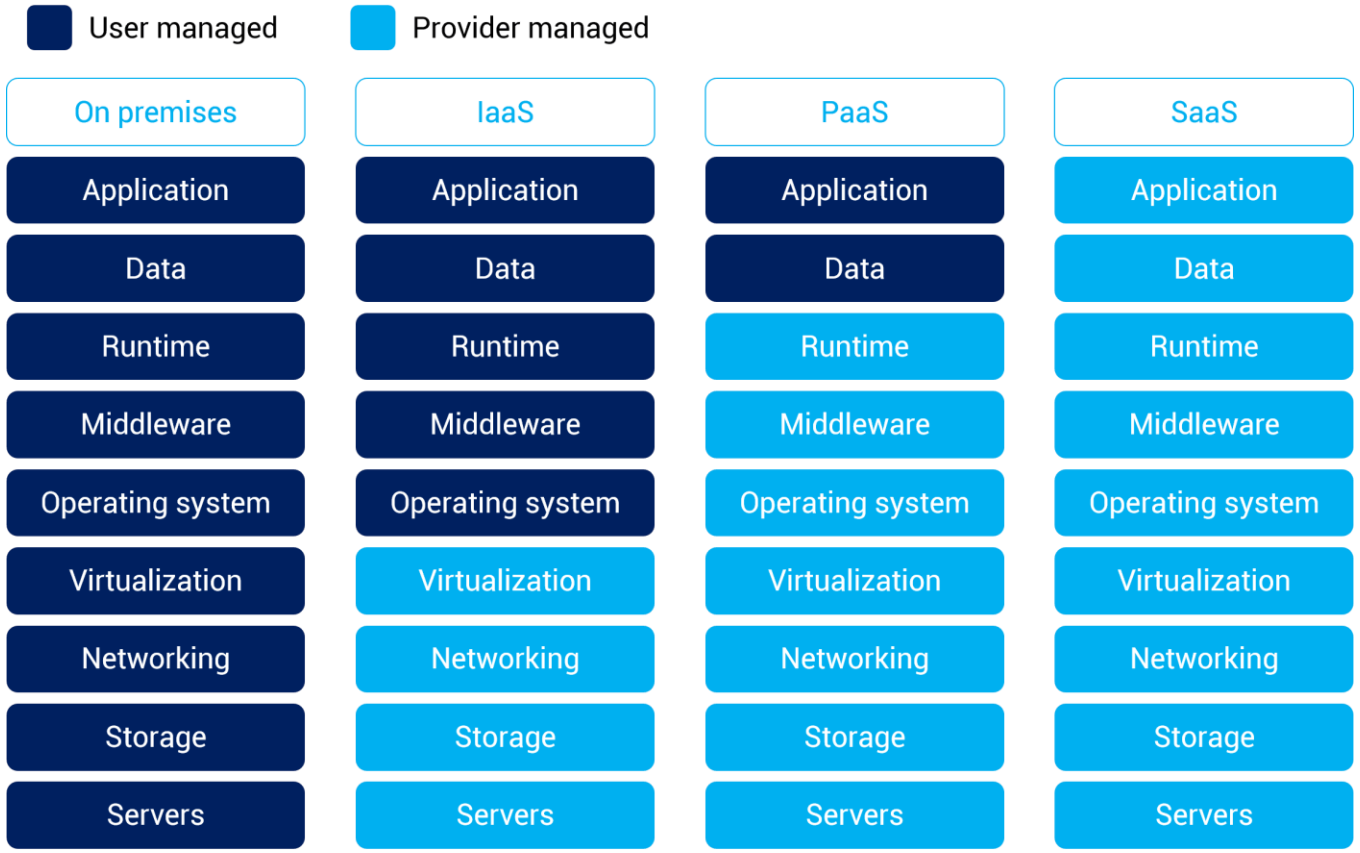
- Kodiranje
- Kontinualnu integraciju (CI)
- Kontinualno testiranje
- Skladištenje verzija softvera (Packaging)
- Puštanje u rad (Release)
- Konfigurisanje
- Monitoring (praćenje zdravlja aplikacije)

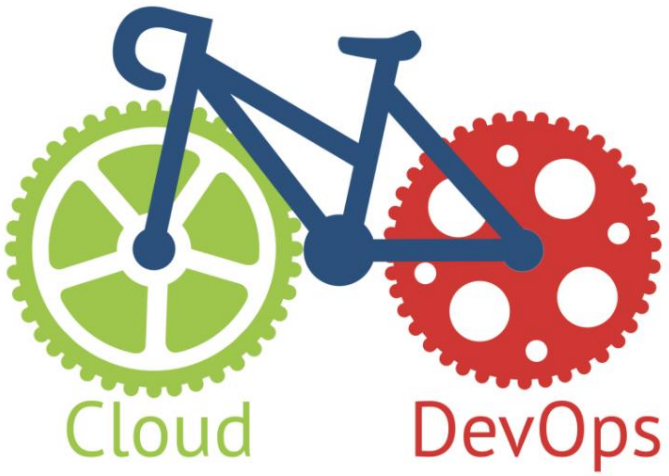
I dalje ostaje pitanje šta je to

- Kako kod dolazi do produkcije (korisnika?)
- Kava je infrastruktura na kojoj se kod razvija?
- Imamo aplikaciju, ali na čemu se ona izvršava (OS, virtualizacija, zavisnosti)?
- Kako se proverava status aplikacije?
- Imamo aplikaciju, koji je sledeći korak u razvoju?



DevOps VS Cloud

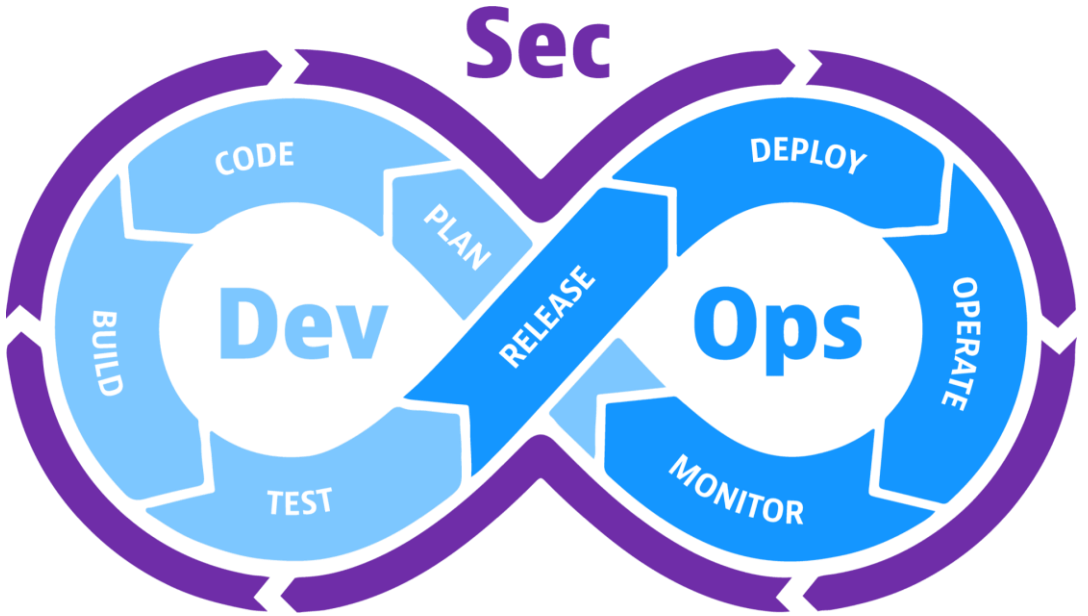




DevSecOps

Klaud usluga može imati vrlo komplikovanu komunikaciju, slojevitost i podelu odgovornosti.
Neophodno je razumeti sve aspekte ovog modela i primeniti adekvatne sigurnosne prakse kako bi se izbeglo kompromitovanje cele strukture.

Neke od praksi: firewall, konfiguracija, izloženost servisa, protokoli…



Rukovanje izuzecima

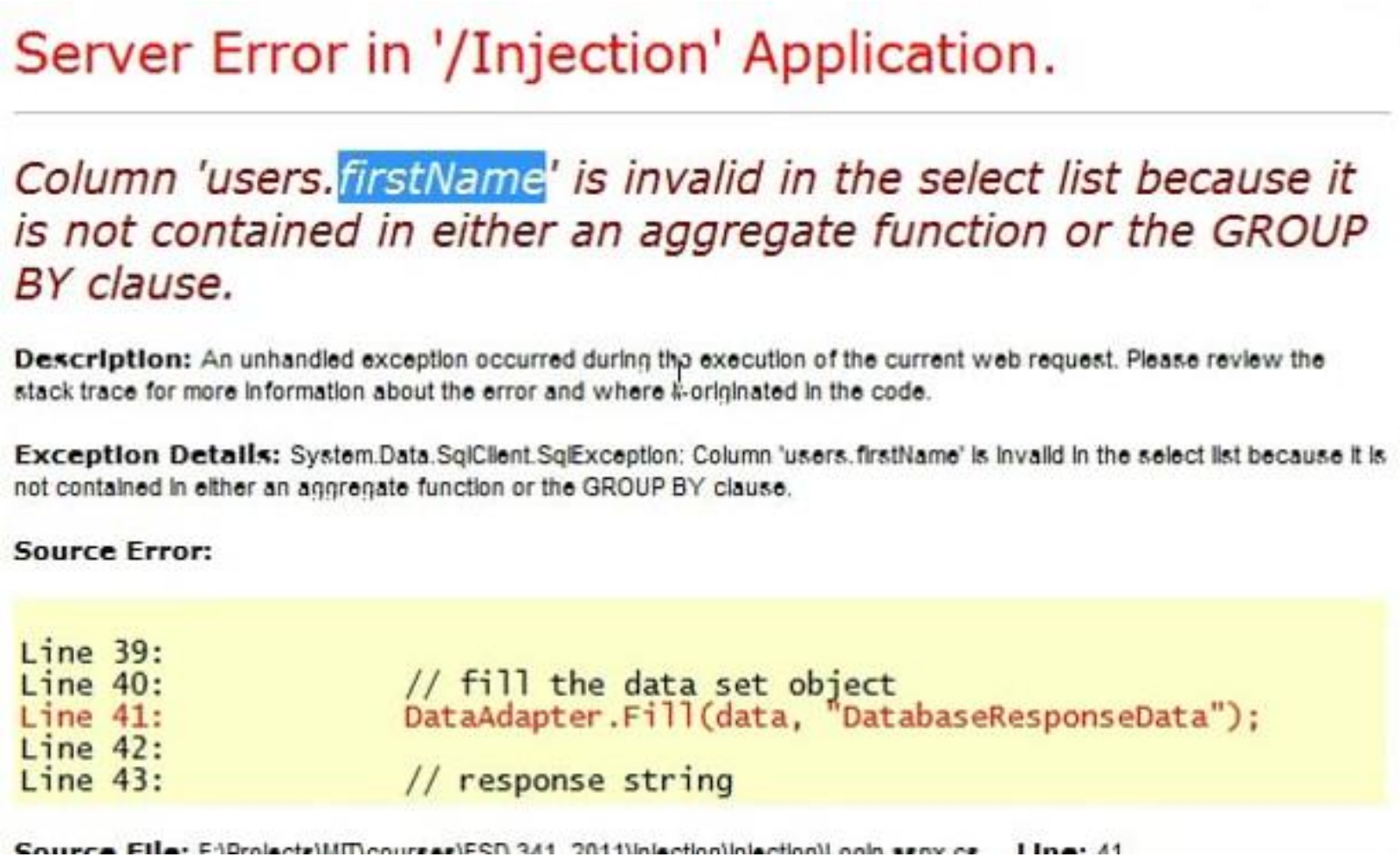
- Deo implementacije (kodiranja)
- Za operaciju softvera je jako važno ispravno rukovanje izuzecima (exception handling)
  - Bolja poruka korisniku šta je pogrešio ili do kakvog je problema došlo
  - Čuvanje zdravlja aplikacije – greške u aplikaciji ne ruše aplikaciju (primer: trenutna nedostupnost konekcije)

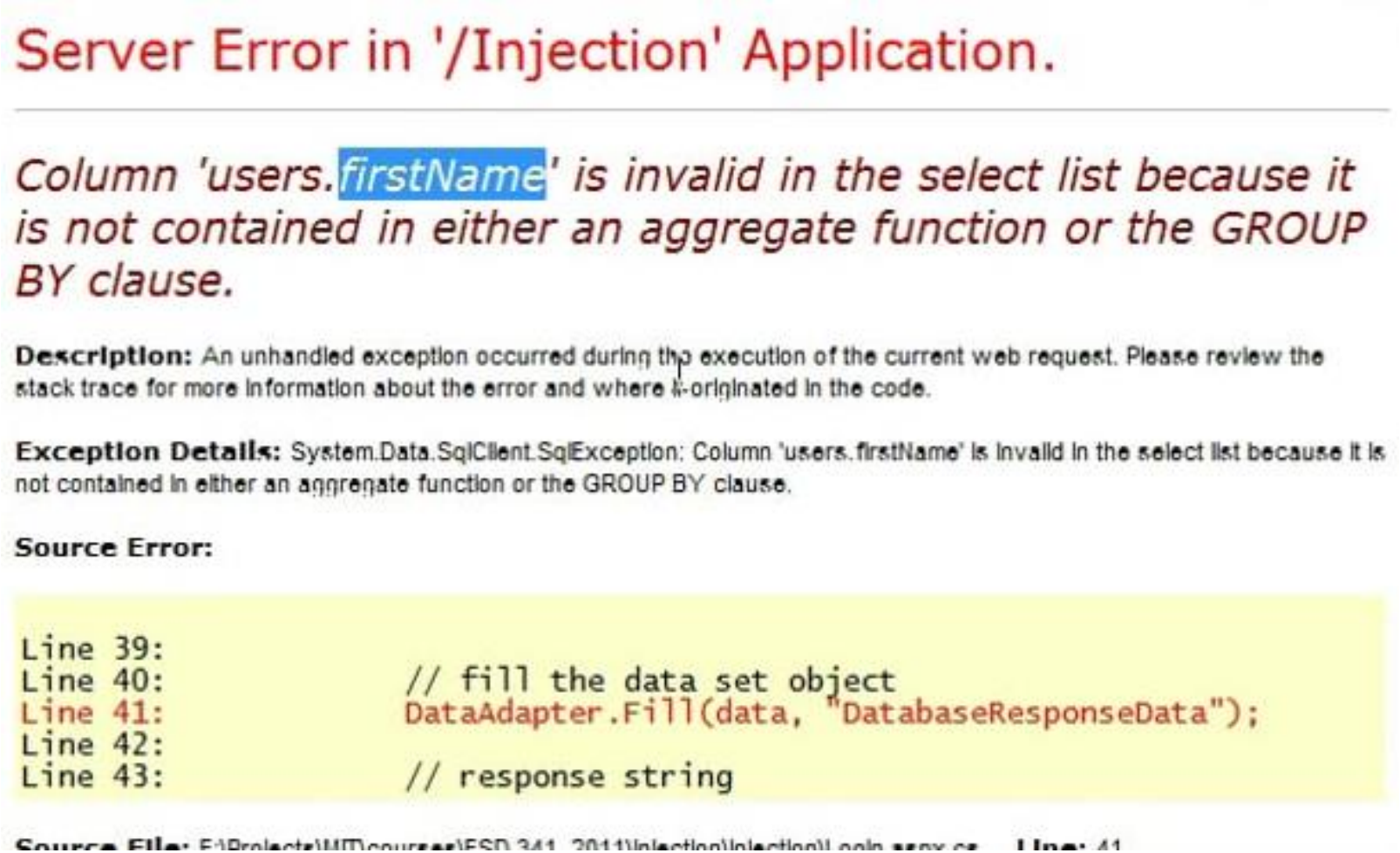
Rukovanje izuzecima

- Zašto je rukovanje izuzecima važno za sigurnost aplikacije?

- Šta napadač može da iskoristi ukoliko nepravilno rukujemo izuzecima?

Izuzeci koji nisu obrađeni mogu otkriti podatke o sistemu!





Olakšali smo posao napadaču. Sada zna:

- Imamo users tabelu
- Tabela ima firstName
- Imamo GROUP BY u upitu
- Konvencija imenovanja je camelCase
- Koristimo ADO.NET
- Koristimo ASP.NET

...

Razmišljajte o izuzecima

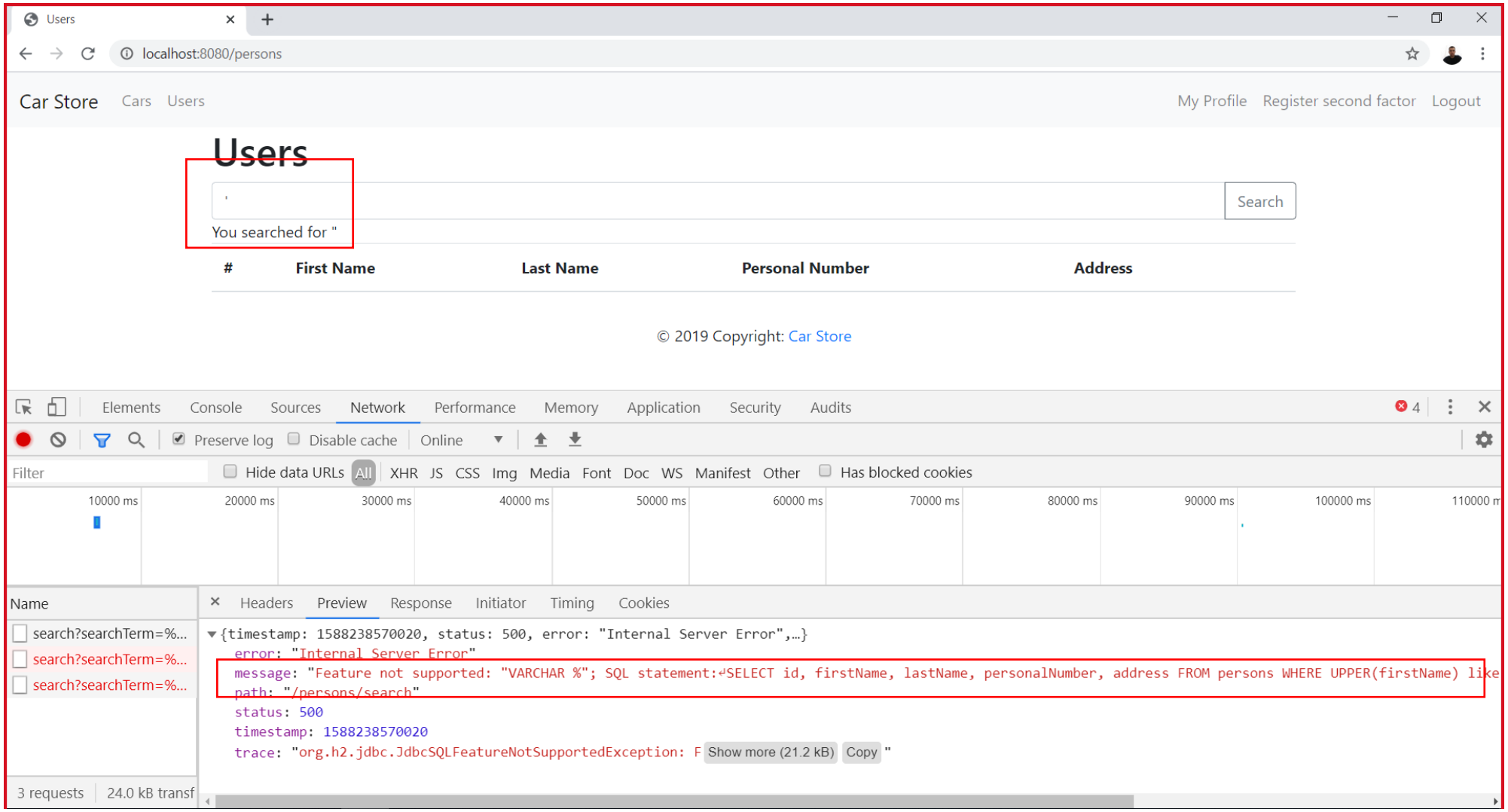
- Koristite try-catch blok za izuzetke koji su lokalizovani i očekivani
- Koristite aspektno orijentisano programiranje (JAVA anotacije, .NET attributi) za izuzetke koji se pojavljuju na više mesta ili su neočekivani

- Primer izuzetaka koji su “cross-cutting”, to jest pojavljuju se na više mesta:
  - Strana nije pronađena -> Error Code 404
  - Server greška -> Error Code 500
  - Nije dozvoljen pristup strani ili resursu -> Error Code 403

Pronalaženje neobrađenih izuzetaka i popravka

Demonstracija na pretrazi korisnika

Demonstracija



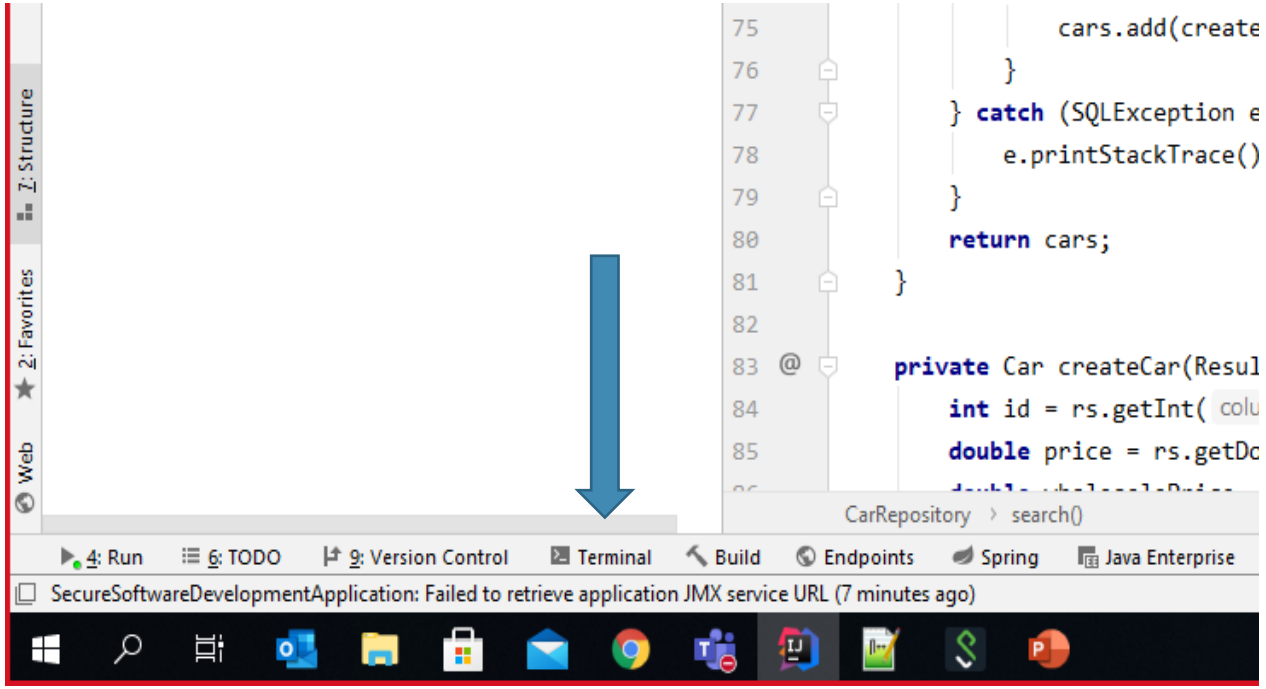
Demonstracija

- Razlog zašto se vraća exception jeste što nemamo catch blok gde se obrađuje
- Metoda *search* koja se koristi prilikom pretrage nalazi se u *PersonRepository* klasi

```
public List<Person> search(String searchTerm) throws SQLException {
    List<Person> personList = new ArrayList<>();
    String query = "SELECT id, firstName, lastName, personalNumber, address FROM persons WHERE
UPPER(firstName) like UPPER('%" + searchTerm + "%')" +
            " OR UPPER(lastName) like UPPER('%" + searchTerm + "%')";
    try (Connection connection = dataSource.getConnection();
         Statement statement = connection.createStatement();
         ResultSet rs = statement.executeQuery(query)) {
        while (rs.next()) {
            personList.add(createPersonFromResultSet(rs));
        }
    }
    return personList;
}
```

Priprema za samostalni rad

1. Otvorite IntelliJ Idea
2. U terminalu izvršite **git checkout master**
3. Zatim **git reset --hard**
4. Pokrenite aplikaciju
5. Prijavite se
   - username: **bruce**
   - password: **wayne**



Samostalni rad

Opis:

- Izazvati izuzetak na stranici detalja o automobilu koji prikazuje korišćeni SQL upit
- Obraditi izuzetak
- Vreme: **15 minuta**

Dodatni zadatak:

- Izvesti SQL injection napad koji briše sve korisnike iz *persons* tabele pomoću informacija dobijenih iz prouzrokovanog izuzetka

Rešenje zadatka

Demonstracija

Logovanje (Logging)

- Čuvanje istorije relevantnih događaja u sistemu
- Logovi se obično čuvaju u tekstualnim fajlovima i sadrže
  - Timestamp
  - Deo sistema (npr. klasa UserService)
  - User ID
  - Kategorizacija
  - Opis događaja

Kategorizacija

- **Error** – kritična (fatalna) greška u sistemu. Nije svaki Exception, već samo onaj koji sistem čini nestabilnim ili je neočekivan.
- **Warning** – Očekivana ili uobičajena greška, izazvana korisničkom akcijom ili eksternim faktorima (npr. konekcija ne postoji). Sistem može da se oporavi od ovakve greške.
- **Info** – Bilo koji događaj od značaja za sistem.
- **Debug** – Svi ostali događaji potrebni za analizu problema u produkciji (live system).

- Što je nivo veći, to je manje takvih događaja.

Opis

- Opis je slobodan tekst
- Treba da bude koncizan i da sadrži relevantne podatke za debugovanje na produkciji
  - ID stranice
  - Korisnički unos
- **Ne treba logovati osetljive podatke**
  - Lozinku
  - JMBG
- Ne treba praviti unos ukoliko nam ne daje relevantne informacije
- Postavite sebe na mesto osobe koja čita logove

Logovanje (Logging)

Olakšava analizu

- Programerskih grešaka u sistemu
- Nepredviđenih grešaka
- Korisničkih grešaka (korisnička podrška)
- Izvor nekonzistentnih podataka u sistemu

Omogućava

- Monitoring – praćenje zdravlja sistema uživo
- Korektivne i preventivne mere kroz periodičnu analizu logova

Spring Logger

Svaka klasa treba da ima svoju instancu loggera koji identifikuje iz koje klase je log unos nastao.

```
private static final Logger LOG =
LoggerFactory.getLogger(UserService.class);
```

Logging i Monitoring

U slučaju napada logovanje je neverovatno korisno.

- Analiza kako je napad izvršen (cyber forensics)
- Prikupljanje dokaza za sudski proces ili istragu (IP adresa...)
- Pronalaženje i popravka ranjivosti

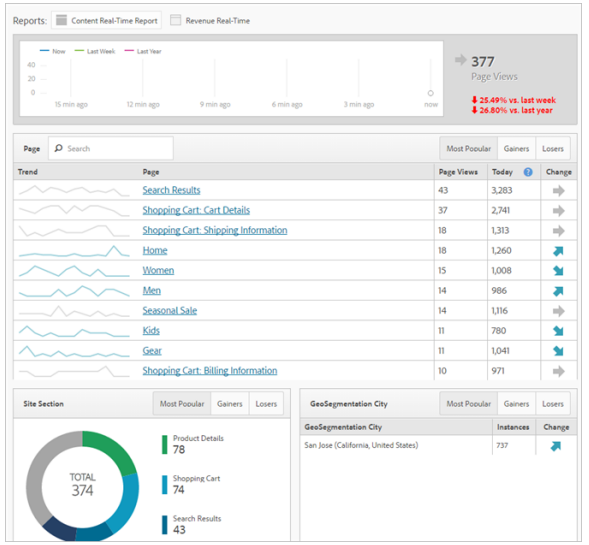
Logovanje pomaže u otklanjanju posledica, monitoring je preventiva.

- Monitoring pomoću sistema za detekciju ili prevenciju intruzije – Intrusion Detection/Prevention System (IDS/IPS)
- Monitoring nam daje razne korisne informacije o stanju sistema

Monitoring

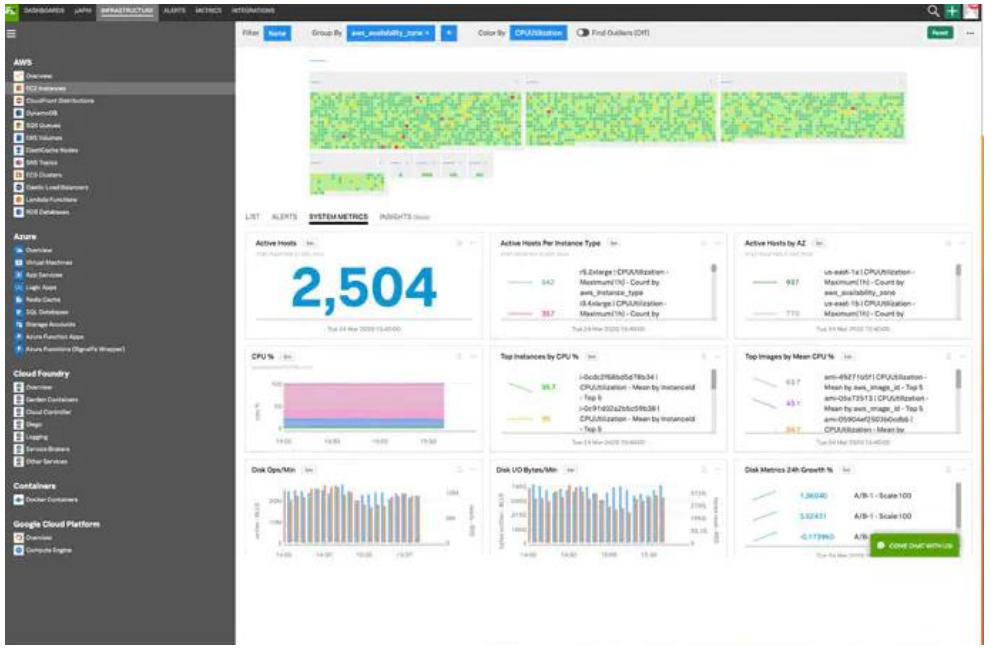
Primeri alata za monitoring:

Adobe Analytics



Prometheus & Grafana

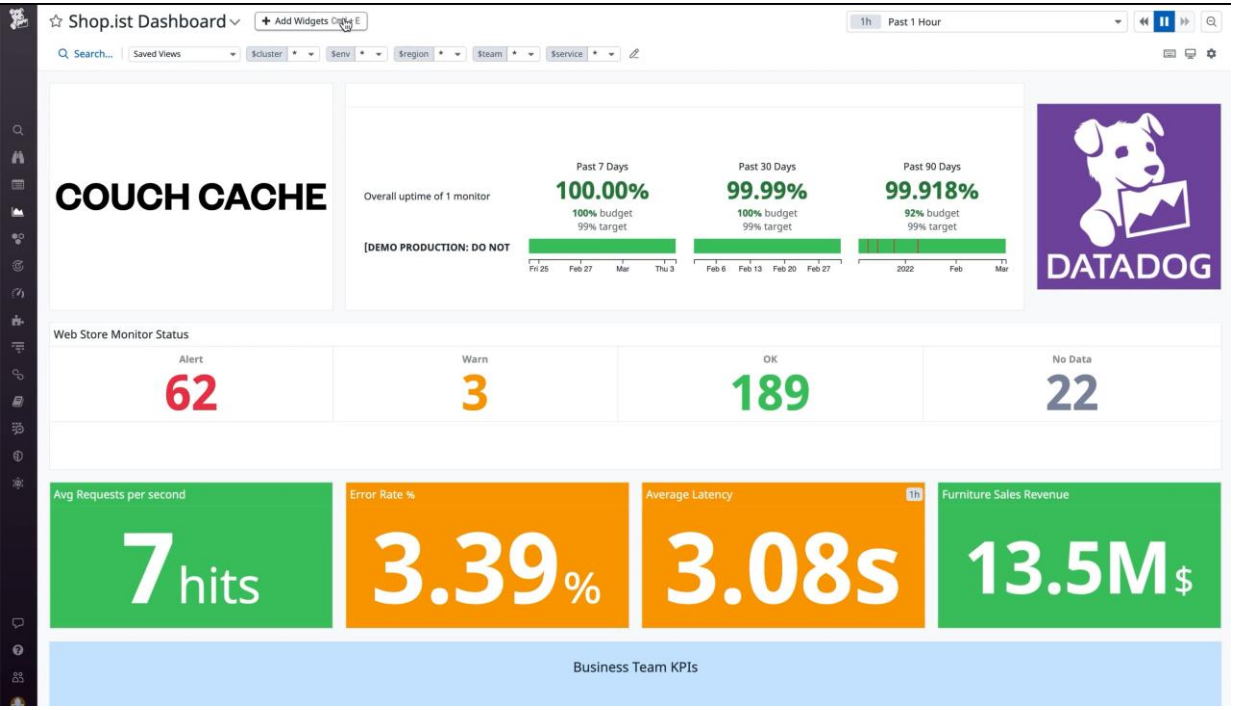
Splunk



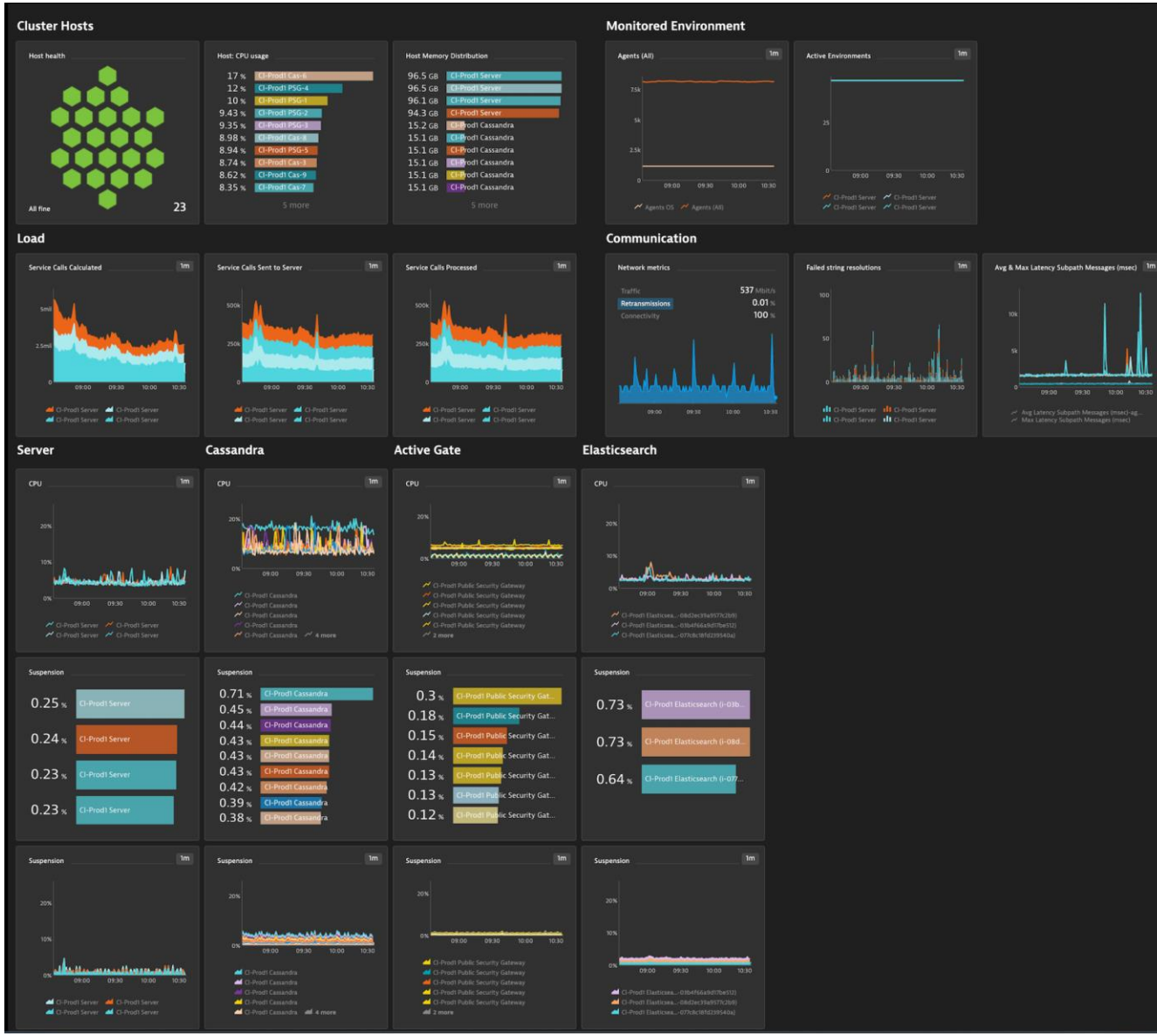
Monitoring

Primeri alata za monitoring:

DataDog



Splunk



Logging i Auditing

- **Auditing** je praćenje događaja koji su od interesa za sigurnost
- Praćenje korisničkih akcija koje su **osetljive**
- Razlog je pružanje sigurnosne usluge **neporecivosti (non-repudiation)**

Logging i Auditing

- Logging i auditing su veoma slični ali se njihova **svrha** razlikuje
- Primer podataka relevantnih prilikom auditinga:
  - Korisnik koji vrši akciju
  - Entitet nad kojim se vrši akcija
  - Pri vršenju promene: stara i nova vrednost

- Primeri:
  - Prijava (uspešna i neuspešna)
  - Transakcija novca
  - Brisanje korisnika

Logging i Auditing

Primena Logging-a:
"Zlatokosa je ušla u šumu."

Logging i Auditing

Primena Auditing-a:
"Neko je polomio moju stolicu!"
"Neko je pojeo moju kašu!"

Logging i Auditing

Demonstracija na pretrazi korisnika

Demonstracija na login korisnika

Demonstracija Logging-a

- U klasi PersonRepository i metodi search:

```
try (Connection connection = dataSource.getConnection();
     Statement statement = connection.createStatement();
     ResultSet rs = statement.executeQuery(query)) {
    while (rs.next()) {
        personList.add(createPersonFromResultSet(rs));
    }
} catch (SQLException ex) {
    LOG.warn("Person search failed for searchTerm " + searchTerm, ex);
}
```

Demonstracija Auditing-a

- U klasi DatabaseAuthenticationProvider i metodi authenticate koja se bavi prijavom korisnika

```
if (success) {
    AuditLogger.getAuditLogger(DatabaseAuthenticationProvider.class).audit("Login 
successful for username '" + username + "'");
    User user = userRepository.findUser(username);
    List<GrantedAuthority> grantedAuthorities = getGrantedAuthorities(user);
    return new UsernamePasswordAuthenticationToken(user, password, grantedAuthorities);
}

AuditLogger.getAuditLogger(DatabaseAuthenticationProvider.class)
        .audit("Login failed for username '" + username + "'");
```

Demonstracija

- Logovi će izgledati ovako:

```
2020-04-30 11:53:47.036  INFO 20736 --- [nio-8080-exec-5]
c.z.s.c.DatabaseAuthenticationProvider   : userId=null - Login failed for username 'asdad’

2020-04-30 11:54:02.488  INFO 20736 --- [nio-8080-exec-4]
c.z.s.c.DatabaseAuthenticationProvider   : userId=null - Login successful for username
'peter’
```

- userId je null jer pre prijave još uvek ne znamo id korisnika

Samostalni rad

Opis:

- Logovati prethodno obrađeni izuzetak (prvi samostalni rad – Edit car) sa odgovarajućom kategorijom i podacima
- Dodati auditing na osetljive operacije na portalu sa odgovarajućim podacima
  - Buy car, JMBG edit, Edit car price
  - Nađite još jednu operaciju modifikacije i primenite auditing
  - Koristite metode **AuditLogger.audit** i **AuditLogger.auditChange**
- Vreme: **15 minuta**

Rešenje zadatka

Demonstracija

Zaključak

Sigurnost

Cilj kursa

Zaštita reputacije organizacije

Sprečavanje finansijske štete

Zaštita korisnikovih podataka

Zaštita krađe znanja (industrijska špijunaža)

Ispunjavanje pravne regulative (zaštita podataka, PCI, medicinski podaci)

➔ 100% sigurnost je mit

Trudimo se da učinimo troškove napada (i rizik) većim nego dobit napadača

Obrađene teme

Tema	Opis
Uvod u kurs i alate	
SQL Injection	Kako identifikovati i popraviti ranjivost koristeći ustanovljene pristupe i frameworke
Cross Site Scripting (XSS)	
Cross Site Request Forgery (CSRF)	
Alati za statičku i dinamičku analizu	SonarQube (statička) OWASP ZAP (dinamička) Tumačenje izveštaja
Sigurna implementacija autentifikacije	Sigurno skladištenje lozinke Dvofaktorska autentifikacija
Autorizacioni modeli	Analiza zahteva i implementacija RBAC
Sigurnost i DevOps	Obrada izuzetaka, logging i auditing

OWASP TOP 10

2021

A01:2021-Broken Access Control

A02:2021-Cryptographic Failures

A03:2021-Injection

A04:2021-Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

Milica Milošević

Danko Miladinović