Zajemanje in vizualizacija agregiranega dnevnika dogodkov sistema Symbiot

Amadej Pavšič[1], Matevž Pustišek[2], Luka Mali[3]

[1]*Fakulteta za elektrotehniko in Fakulteta za računalništvo in informatiko, Univerza v Ljubljani*
[2,3]*Fakulteta za elektrotehniko, Univerza v Ljubljani*
E-pošta: [1]ap5512@st.udent.uni-lj.si, [2]matevz.pustisek@fe.uni-lj.si, [3]luka.mali@fe.uni-lj.si

Capturing and visualizing the aggregated event log of the Symbiot system

The article defines the problem of displaying a specific Symbiot system aggregated event log and seeks a solution for it. The main problem of the Symbiot system in the area of event log display is not having one overall view of all event logs of the system and thus its inability to locate and drill-down any errors in the system.

With defined requirements and use-case, we have researched technologies to create an integrated solution of event log visualization. The scope of the solution includes the data flow and the display of the data on the web interface. Using our research results, two proof-of-concept solutions have been made, using two open-source data observation platforms - Grafana and Kibana.

Comparing the two solutions, we concluded, that the Kibana platform emphasises data exploration, while the Grafana platform emphasises a modular interface construction that serves as a final solution presented to the client. Based on the specific requirements of the task, we concluded that the concept of the Grafana solution, due to its data use philosophy and the individual components that suit our specific use case, is more appropriate.

1 Uvod

V članku smo se osredotočili na dnevniške zapise dogodkov (ang. *event logs*, v nadaljevanju tudi: „dnevnik dogodkov“, „dnevniški zapisi“). Dnevnik dogodkov je, na kratko, zbirka zapisov dogodkov, ki nam z raznimi podatki (kot so časovni žig, vir dogodka, nivo resnosti, sporočilo dogodka ipd.) povedo, kaj se na posameznih komponentah nekega programskega sistema dogaja [1].

Z vedno večjimi in bolj kompliciranimi programskimi sistemi se veča tudi število operacij in dogodkov. Če želimo tak sistem razumeti in ohranjati njegovo varnost, so dnevniki dogodkov nepogrešljiva pomoč. Ker je spremljanje ter odpravljanje morebitnih napak na sistemu veliko lažje z grafičnim uporabniškim vmesnikom, ki prikazuje zapise dogodkov za trenutni in pretekli čas, je smiselno ustvarjati vizualizacije dnevnikov dogodkov sistemov.

Ker je naloga nastala v sodelovanju s podjetjem, ki je avtorjev štipenditor, se osredotoča specifično na prikaz določenega dnevnika dogodkov sistema Symbiot podjetja Iskraemeco. S tem naloga ne predstavlja le raziskave področja, temveč ponuja dejanski primer problema, ki ga je treba rešiti s praktičnim konceptom rešitve, ki dokazuje uporabnost raziskanih tehnologij. Namen praktične rešitve je izboljšava trenutnega sistema prikaza dnevnika dogodkov in izpolnitev dejanskih zahtev strank in arhitektov sistema Symbiot.

2 Sistem Symbiot

Sistem Symbiot, produkt podjetja Iskraemeco, je adaptivna in pametna programska platforma, ki omogoča enostavno, varno in avtomatizirano upravljanje podatkov v realnem času [2]. Primarno je uporabljena za povezovanje električnih števcev v t. i. pametno omrežje (ang. *smart grid*) z arhitekturo AMI (ang. *Advanced Meter Infrastructure*).

Arhitektura AMI predvideva dve glavni komponenti sistema – MDM (ang. *Meter Data Management*) in HES (ang. *Head-End System*), kar omogoča dvosmerno komunikacijo med pametnimi števci in programsko platformo ter s tem prejemanje in procesiranje podatkov v realnem času, kar posledično pomeni efektivnejše in pametnejše upravljanje s celotnim sistemom [3]. Sistem Symbiot je zgrajen modularno, tako da ima vsaka od teh dveh komponent polno prilagodljivih različnih storitev, ki opravljajo specifične naloge – povezovanje z napravami, branje podatkov iz naprav, pridobivanje podatkov iz podatkovnih baz itd.

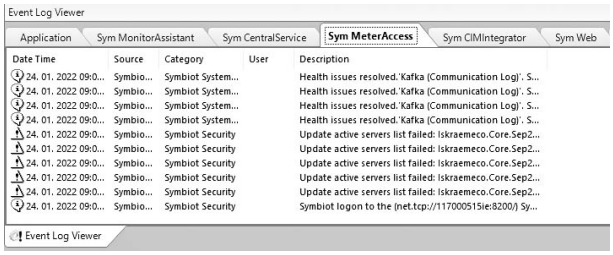
2.1 Trenutno stanje sistema

Uporabniki sistem primarno upravljajo preko namiznega programa Symbiot Configurator, ki omogoča podrobno upravljanje z vsako komponento sistema posebej. Prav tako ponuja pregled ter dodajanje baz podatkov, naprav in ostalih komponent ter vrsto drugih funkcionalnosti.

Vse komponente v sistemu Symbiot informacije o izvršenih nalogah zapisujejo v dnevnik dogodkov. Zaenkrat so dnevniški zapisi dogodkov prikazani ločeno po storitvah in vidni le z različico Windowsovega programa Event Log Viewer znotraj aplikacije Symbiot Configurator, prikazano na sliki 1.

2.2 Želeno stanje sistema

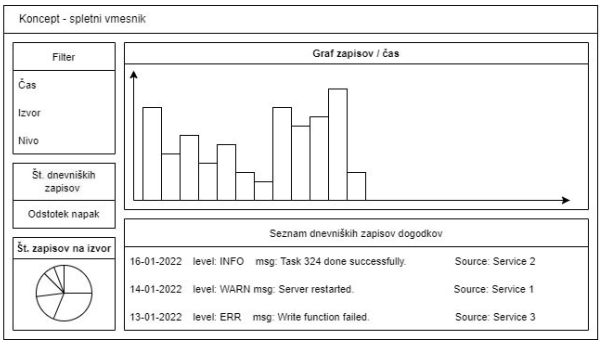
Znotraj podjetja se je razvila rešitev agregiranja dnevnikov dogodkov posameznih storitev in so dostopni na enem viru. Ideja je, da se ta združen dnevnik dogodkov



Slika 1: Prikazan dnevnik dogodkov storitve Symbiot MeterAccess. Vidni so tudi zavihki z nekaterimi ostalimi storitvami.

sistema Symbiot vizualizira na nadzorni plošči, ki uporabnikom omogoča enostaven pregled nad celotnim sistemom in predvsem lažje odkrivanje in odpravljanje težav.

Definirali smo tudi zahteve tovrstne plošče in jih grafično prikazali na skici 2. Glavni komponenti sta grafični prikaz števila napak po času ter tekstualni prikaz v obliki seznama dnevniških zapisov dogodkov, ki se ujemajo s časovnim razponom grafa. Poleg tega je nujno filtriranje dogodkov glede na čas, izvor in nivo resnosti. Opcijski pa so še dodatni številčni prikazi v pomoč, npr. odstotek napak in število zapisov glede na izvor.



Slika 2: Skica želenega uporabniškega vmesnika z zahtevanimi komponentami.

2.3 Primer uporabe

Z našo rešitvijo tako želimo izpolniti sledeči primer uporabe (ang. *use-case*). Uporabnik opazi, da sistem Symbiot ne deluje, kot bi moral, zato odpre nadzorno ploščo s prikazanim agregiranim dnevnikom dogodkov. Na plošči izbere časovni interval zadnjih treh ur in na grafu takoj opazi anomalijo – izbruh napak. Časovni interval omeji samo na to špico napak in dogodke filtrira samo po nivoju resnosti napake. Prestavi se na seznam dogodkov, kjer lahko po kronološko razporejenih dogodkih išče izstopajočega. Ko na seznamu klikne na posamezni dogodek, se mu prikažejo podrobnosti le-tega. Tako poišče izvor problema, torej tisti dogodek ali skupino njih, ki so sprožili celoten izbruh napak.

3 Tehnologije prikazovanja dnevnika dogodkov

Tekom naloge smo raziskali tehnologije, ki se navezujejo na področje celostne rešitve prikazovanja dnevnika dogodkov, kar obsega zaledni (ang. *backend*) in čelni (ang. *frontend*) del. Na podlagi ugotovitev smo izbrali najustreznejše posamezne tehnologije, ki bodo predstavljale posamezne komponente za gradnjo celotne rešitve.

3.1 Platforme za vizualizacijo dnevnikov dogodkov

Sploh v zadnjih letih se razvija veliko platform za vizualizacijo dnevniških zapisov dogodkov. Vsaka se osredotoča na posamezni problem gradnje tovrstnih rešitev, ampak v splošnem sta zelo priporočani odprtokodni rešitvi Grafana in Kibana, zato smo se tudi sami odločili za podrobnejši pregled le-teh. Poleg tega imata obe tudi številno in aktivno skupnost, v kateri je bil preizkušen in rešen že marsikateri problem, kar precej olajša reševanje ovir pri gradnji lastne rešitve.

3.1.1 Grafana Loki sklad

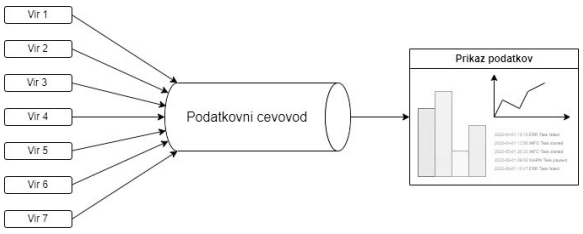
Grafana je celostna rešitev opazovanja in analize podatkov iz katerekoli baze podatkov [4]. Primarno se osredotoča na prikaz merjenih podatkov, ampak ponuja tudi možnosti prikaza dnevnikov dogodkov. Velika prednost Grafane je možnost integracije z mnogo različnimi zalednimi enotami za shranjevanje podatkov. Skupaj z Grafano lahko uporabimo sistem znotraj Grafana sklada, specifično ustvarjen za reševanje zaledja dnevniških zapisov dogodkov, Grafana Loki [5]. Loki poskrbi za indeksiranje in shranjevanje podatkov v lokalno bazo, da ob poizvedbi iz spletnega vmesnika Grafana vrne ustrezen rezultat.

3.1.2 Kibana in Elastic sklad

Kibana pa je samo čelni del celotnega t. i. Elastic sklada (ang. *Elastic stack*) [6]. Za svoje delovanje tako nujno potrebuje sistem Elasticsearch, ki predstavlja univerzalni zaledni del in v našem primeru poskrbi tudi za indeksiranje in shranjevanje podatkov. Je pa Elastic sklad že primarno usmerjen v prikazovanje dnevnikov dogodkov in se v prikaz podatkov drugih vrst usmerja šele zadnja leta.

3.2 Podatkovni tok

Da bi se podatki ustrezno prikazovali, moramo poskrbeti tudi za primeren dotok podatkov do platforme za vizualizacijo. Na splošno je proces toka podatkov pri podobnih sistemih sestavljen iz: 1. virov, ki podatke ustvarijo; 2. podatkovnega cevovoda, ki podatke združi in obdela; 3. ponorov, ki podatke uporabijo za nadaljnjo uporabo.



Slika 3: Splošni diagram toka podatkov v kontekstu prikazovanja velike količine podatkov.

Slednji proces je prikazan tudi s shemo na Sliki 3. Za naš primer so ti viri razne komponente sistema Symbiot, torej od električnih števcev do raznih Windows storitev,

baz podatkov itn. Ponor podatkov pa bo spletni vmesnik z vizualizacijo podatkov Grafana oziroma Kibana. Vsi procesi med njimi spadajo pod pojem podatkovnega cevovoda, ki ga lahko definiramo kot „skupek procesov, ki premikajo in preoblikujejo podatke iz raznih virov do nekega cilja, kjer se iz podatkov pridobi nova znanja in informacije“ [7].

3.2.1 Apache Kafka

Za združevanje podatkov iz različnih virov poskrbi že zunanja rešitev podjetja. Za našo nalogo smo imeli efektivno en vir podatkov v obliki Kafka teme, zapisi dogodkov pa so v JSON obliki. Apache Kafka je odprtokodna platforma za porazdeljeno pretakanje dogodkov [8]. Konceptualno je dogodek za Kafko sporočilo z določenim časovnim žigom, ključem ter vrednostjo in dodatnimi metapodatki. Kafka se izvaja v skupku strežnikov in se tako enostavno horizontalno razširi. Dogodki se organizirajo in shranjujejo v t. i. temo (ang. *topic*), ki jo lahko poenostavljeno enačimo z mapo v datotečnem sistemu, datoteke znotraj mape pa predstavljajo posamezne dogodke [8].

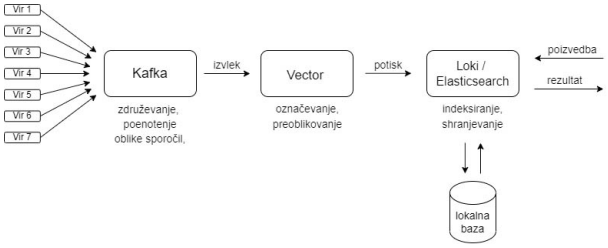
3.2.2 Posrednik podatkov

Ker direktna povezava Kafka teme z Grafano ali Kibano ni bila mogoča, je bil potreben še posrednik podatkov. Kljub temu da imata sklada obeh omenjenih platform tudi svoje posrednike, smo izbrali eno univerzalno orodje za posredovanje podatkov, Vector [9]. Izraz posrednika je lahko podcenjujoč, saj ima v primeru predstavljene arhitekturne zasnove podatkovnega cevovoda le-ta osrednjo nalogo. Posrednik Vector, predstavljen tukaj, je pravzaprav „uživalec podatkov“ (ang. *data ingester*), torej skrbi za ekstrahiranje in nalaganje (ali „izvlek in potisk“) podatkov. Poleg uživanja podatkov skrbi tudi za lahko transformacijo podatkov v obliki poenotenja oblike in označevanja podatkov.

3.2.3 Zasnova in struktura podatkovnega cevovoda

Sedaj lahko s Sliko 4 tudi podrobneje predstavimo, kako za naš primer izgleda podatkovni tok. Cevovod ima lahko različne strukture. Najbolj uporabljena sta ELT in ETL struktura, kjer črke v kratici pomenijo E – ekstrakcija (ang. *extraction*), L – nalaganje (ang. *loading*) in T – transformacija (ang. *transformation*). Struktura nam tako pove vrstni red, v katerem se omenjene operacije nad podatki izvajajo.

Pri ekstrakciji podatkov si delo razdelita Kafka in Vector, saj slednji izvleče podatke iz Kafkine teme, kjer so agregirani podatki tudi že ekstrahirani iz drugih zunanjih virov. Potem sledi prva, t.i. „lahka“, transformacija znotraj orodja Vector, nato pa se podatki potisnejo (oziroma „naložijo“, kar predstavlja „L“ del v strukturi) v Loki oziroma Elasticsearch, kjer pa se še enkrat in bolj znatno transformirajo, ko jih omenjeni orodji skladno s svojo sistemsko zgradbo indeksirata in shranita. Zaradi te dvojne transformacije je struktura našega cevovoda malo bolj specifična in se imenuje EtLT [10], kjer „t“ predstavlja omenjeno lahko transformacijo na posredniku Vector.



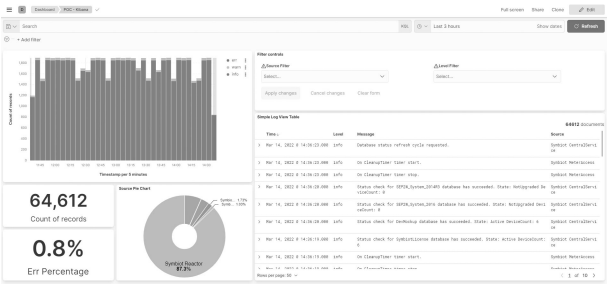
Slika 4: Podrobni prikaz podatkovnega toka za naš primer.

4 Gradnja konceptnih rešitev

Zavoljo čim realnejše primerjave omenjenih tehnologij je bilo najbolje narediti dve rešitvi, ki se ju je nato med seboj primerjalo. Postavitev in konfiguracija posrednika podatkov Vector ter sistemov Loki in Elasticsearch je bila precej enostavna, potrebno je bilo samo fino uglaševanje.

Gradnja spletnih vmesnikov na obeh platformah Grafana in Kibana temelji na grafičnih vsebnikih ali vstavkih (ang. *panel*). Končni nadzorni plošči sta prikazani na Slikah 5 in 6. Tudi tukaj z gradnjo ni bilo večjih težav, saj sta obe platformi zelo intuitivni za uporabo.

Slika 5: Celotna rešitev s spletnim vmesnikom Grafana.



Slika 6: Celotna rešitev s spletnim vmesnikom Kibana.

5 Primerjava rešitev

Obe rešitvi zadovoljujeta zahteve in rešujeta primer uporabe, kljub temu pa je med končnima rešitvama nekaj opaznih razlik.

5.1 Primerjava uporabniških vmesnikov

Prva večja razlika je v grafičnem prikazu števila zapisov glede na čas. Pri Kibani je graf t. i. naložen stolpčni grafikon in je za naš primer prikaza zapisov dogodkov

s tremi različnimi resnostmi zelo pregleden. Pri Grafani tak graf ni bil mogoč, zato so zapisi prikazani s črtnim grafikonom.

Druga opazna razlika pa je v seznamu dnevniških zapisov dogodkov. Pri Kibani se seznam enostavno generira, ampak nima naprednejših možnosti ali obarvanosti glede na nivo resnosti dogodkov. Pri Grafani ima seznam že privzeto obarvane dogodke, ampak ni pa dobro prilagojen prikazovanju JSON oblike podatkov, zato smo preizkusili še ustvarjanje lastnega vstavka v Grafani.

5.2 Ustvarjanje lastnega vstavka v Grafani

Grafana ponuja dokaj obsežno dokumentacijo za ustvarjanje lastnih grafičnih vsebnikov, ki se jih nato enostavno doda na nadzorno ploščo. Tudi gradnja vstavka nekomu z znanjem React okvirja ter jezika Typescript ne predstavlja večjih težav. Tako smo ustvarili seznam dnevnika dogodkov, viden na Sliki 5, ki je tako pregleden kot prilagodljiv. Kibana tovrstnega ustvarjanja ne podpira tako kot Grafana.

5.3 Primerjava pristopov k prikazovanju podatkov

Ob gradnji smo prišli do zaključka, da se platformi osredotočata na različna načina uporabe spletnih vmesnikov. Pri Grafani je večji poudarek na posameznih vstavkih — vsak vstavek ima svojo poizvedbo, lahko tudi iz različnih virov. Prav tako se podatke lahko filtrira za vsak vstavek posebej. Medtem ima pregledna plošča pri Kibani načeloma samo eno glavno poizvedbo in tudi filtriranje se načeloma izvede na vseh vstavkih na plošči. Uporaba Kibane je tako bolj usmerjena v sprotno raziskovanje podatkov, z rednim spreminjanjem poizvedbe in opazovanjem relacij na enem viru, Grafana pa bolj spodbuja gradnjo plošče kot neke zaključene rešitve, z jasneje definiranimi poizvedbami in z vnaprej določenimi vhodnimi podatki, kar nam za naš primer izgradnje vmesnika za stranko bolj ustreza.

5.4 Primerjava učinkovitosti

Loki in Elasticsearch se razlikujeta v načinu indeksiranja podatkov. To je najbolj vidno pri učinkovitosti sistemov. S testiranjem smo prišli do zaključka, ki je bil glede na indeksiranje tudi pričakovan – Grafana porabi veliko manj pomnilnika, procesorja in predvsem prostora na disku, ampak za ceno počasnejšega nalaganja poizvedb. Kibana ima v primerjavi z Grafano praktično takojšnje čase nalaganja, ne glede na velikost poizvedbe, ampak porabi temu primerno več virov in prostora. Z vidika učinkovitosti smo za ustreznejšo platformo določili Grafano, samo zaradi dejstva, da nam glede na zahteve podjetja in strank več pomenijo prosti računalniški viri.

5.5 Možne izboljšave

Ena od možnih izboljšav je obogatitev rešitve z umetno inteligenco. Obe platformi uporabnikom s plačljivimi naročninami ponujata grajenje strojno naučenih modelov že kar preko spletnega vmesnika. Kibana se osredotoča bolj na zaznavanje anomalij v dnevniku dogodkov, medtem ko Grafana kot platforma za prikaze meritev ponuja tudi naprednejše modele z napovedovanjem podatkov ter dinamičnim opozarjanjem. Mogoče pa je tudi ustvariti svoj model, ga na spletni vmesnik pošiljati kot zunanji vir in prikazovati skupaj z dejanskimi podatki. To bi bila smiselna nadgradnja obeh rešitev, ki predstavlja še en korak naprej v prihodnost vizualizacije dnevnika dogodkov in daje dodano vrednost sistemu.

6 Zaključek

V nalogi smo raziskali rešitve prikazovanja dnevnika dogodkov, ki se v zadnjem času zelo razvijajo in so vedno bolj v uporabi. Ob podrobnejši raziskavi in gradnji konceptne rešitve z dvema odprtokodnima platformama, Grafano in Kibano, smo lahko spoznali, da je razvoj rešitev relativno enostaven ter pri obeh zelo podoben. Vseeno je takoj opazno, da je vsak sistem narejen za reševanje določenih problemov, kar se odseva v filozofiji vmesnikov in načinu uporabe.

Na podlagi teh razlik smo prišli do zaključka, da je sklad Grafane primernejši za specifično zadan problem, saj nam kot razvijalcem omogoča lažji razvoj eksaktne rešitve ter hkrati dovoljuje veliko fleksibilnosti za vse specifične potrebe strank. Glavni namen koncepta rešitve – omogočiti uporabniku vmesnik, da enostavno, pregledno in učinkovito pregleduje dnevnik dogodkov ter da ob pojavitvi napak vzrok poišče na enostaven način – je bil dosežen pri obeh rešitvah.

Literatura

[1] A. Chuvakin *et al.*, *Logging and log management: the authoritative guide to understanding the concepts surrounding logging and log management*, ch. 1. Elsevier/Syngress, 2013.

[2] Iskraemeco, "Symbiot." Dosegljivo na: `https://symbiot.iskraemeco.com/`. [Dostopano: 19. 1. 2022].

[3] T. N. Le *et al.*, "Advanced Metering Infrastructure Based on Smart Meters in Smart Grid," v *Smart Metering Technology and Services - Inspirations for Energy Utilities*, IntechOpen, 2016.

[4] Grafana Labs, "Grafana: The open observability platform." Dosegljivo na: `https://grafana.com`. [Dostopano: 3. 1. 2022].

[5] Grafana Labs, "Grafana loki." Github, Dosegljivo na: `https://github.com/grafana/loki`. [Dostopano: 26. 1. 2022].

[6] Elasticsearch B.V., "Elastic sklad." Dosegljivo na: `https://www.elastic.co`. [Dostopano: 31. 1. 2022].

[7] J. Densmore, *Data Pipelines Pocket Reference*, ch. 1. O'Reilly Media, Inc., 2021.

[8] Apache Software Foundation, "Kafka 3.1 Documentation." Dosegljivo na: `https://kafka.apache.org/documentation/`. [Dostopano: 23. 2. 2022].

[9] Datadog, Inc., "Vector." Dosegljivo na: `https://vector.dev/`. [Dostopano: 14. 2. 2022].

[10] J. Densmore, *Data Pipelines Pocket Reference*, ch. 3. O'Reilly Media, Inc., 2021.