UNIVERSITETI I PRISHTINËS

FAKULTETI I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE

PUNIM SEMINARIK

Lënda: Praktika profesionale

Tema: Mbledhja e të dhënave përmes Javascript

Mentori:
Prof.Asoc.Dr Valon Raça

Kadidati:
Valdrin Ejupi

Prishtinë, Korrik 2023

Abstrakt

Ky punim përshkruan një pjesë të punës time praktike në kompaninë Behamics. Behamics është një agjenci krijuese e cila ofron zgjidhje inovative të problemit që shumë dyqane online ballafaqohen që njihet si “shopping cart abandonments”.

Në këtë punim do shpalosë njohuritë dhe përvojën e fituar, si dhe do fokusohem në arritjet si praktikant në pozitën front-end. Përgjatë praktikës jam njoftuar me teknologji/platforma te reja, gjuhë programuese dhe korniza punuese. Gjatë kësaj periudhe kam punuar kryesisht duke shfrytëzuar gjuhën programuese Javascript. Algoritmi që Behamics ka zhvilluar është i shkruar i gjithi në gjuhën programuese Javascript, duke mos përdorur librari të tjera që janë të zhvilluar nga individ/kompani të tjerë.

Algoritmi kryesisht mbledh të dhëna të ndryshme kryesisht sjelljet(ndërveprimet) e përdoruesit me dyqanin përgjatë kohës së tyre në dyqan, me qëllim kryesor të mbledhjes sa më shumë të të dhënave që në fund të bëhet studimi dhe analiza me anë të inteligjencës artificiale të kuptohen arsyet se përse disa përdorues në prag të blerjes vendosin që të lënë dyqanin dhe mos të procedojnë tutje.

Jam shumë i lumtur me rezultatet e kësaj praktike edhe pse ishte një sfidë e re, fusha në të cilën kam punuar ishte e njohur për mua, shkak i njohurive që kam fituar gjatë studimeve sa i përket gjuhës programuese Javascript, ku kjo ka qenë një ndihmesë mjaft e madhe.

Fjalët kyçe: Shopping cart abandonments, Javascript, Algoritmi, Inteligjenca artificiale.

Përmbajtja

Figurat dhe Tabelat

1. Konceptet Themelore

Në mënyrë që të mund t'i përshkruajmë detajet më të avancuara dhe më specifike nga platforma, fillimisht do të paraqesim një pasqyrë të përgjithshme të skriptës së mbledhjes së të dhënave behamics.js.

Behamics.js është një skriptë e shkruar në gjuhën programuese Javascript, e integruar në dyqanin e klientit kjo skriptë ekzekutohet në mënyre automatike pa nevojë të ndërveprimit nga klienti/ja, duke komunikuar kështu direkt me përdoruesit në mënyrë që t'iu shmangen vendimit të largimit nga dyqani, këtë Behamics e arrin duke implementuar mesazhe mbështetëse të cilat janë rezultat i algoritmit të tyre.

Skripta behamics.js ka vetëm një komponent kryesor objektin behamics, ky objekt përmban të gjitha metodat të cilat nevojiten që të ekzekutohen dhe të mundësoj funksionalitetin e kërkuar.

```
behamics = {

   variables:
   {
       userStorageKey: "_bhm_user_id",
       cssPreload: !1,
       page:null,
       hostObject:null,
       behamicsApiUrl: "https://engine.behamics.com",
       target:"vzi",
   },

   user: null,
```

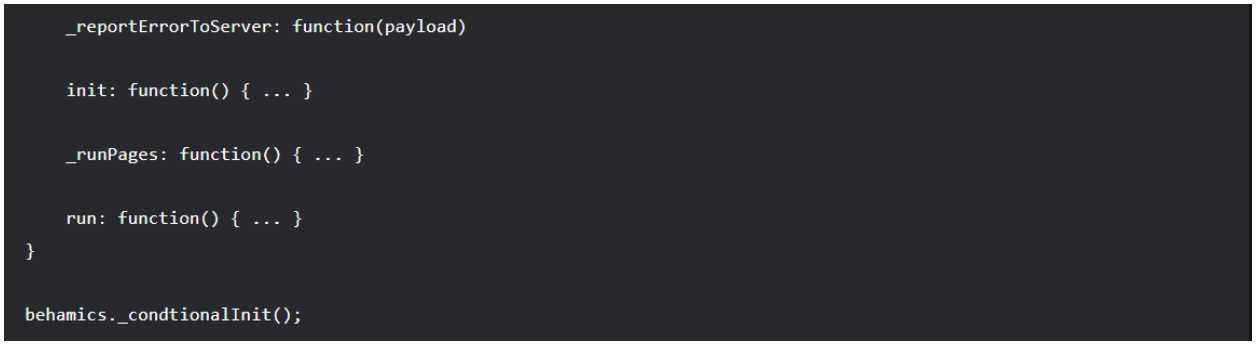

Figura 1: Objekti behamics

Skripta përmban metoda të ndryshme për funksione të caktuara por të gjitha janë të inicializuar brenda objektit behamics.
Jo çdo klientë do ketë metodat e ngjashme, dyqanet e disa klientëve kanë të implementuar shërbime shtesë si listë të dëshirave(wishlist-a), po ashtu janë të ndërtuar nga teknologji të ndryshme, gjuhë programuese të ndryshme kështu skripta do ndryshoj varësisht se qfarë të dhënash vendosim të mbledhim.

Vlen të veçojmë mbledhjen e të dhënave: statike dhe dinamike.
Të dhënat statike i'u referohemi të dhënave që mund ti mbledhim nga DOM-i i dyqanit përmes selektorëve të javascript si: querySelector, getElementByID, etj.
Të dhënat dinamike i'u referohemi interaksionit të përdoruesit me dyqanin, ndryshimeve që përdoruesi shkakton duke klikuar me anë të miut, për këtë implementojm metoda të veqanta të cilët i dëgjojnë këto ndryshime apo evente e që në javascript njihen si event listeners, qëllimi kryesor është mbledhja e saktë e të dhënave.

Këto të dhëna përfshihen në një objekt getmessage te cilin mund te shohim përmes developer tools të ueb shfletuesit përkatësisht nën opsionin network, me pas ky objekt i bashkohet objektit final me emrin finalFormat para se të dërgohet në server ku me pas ruhen në databazë në qoftë se gjithqka është në rregull.

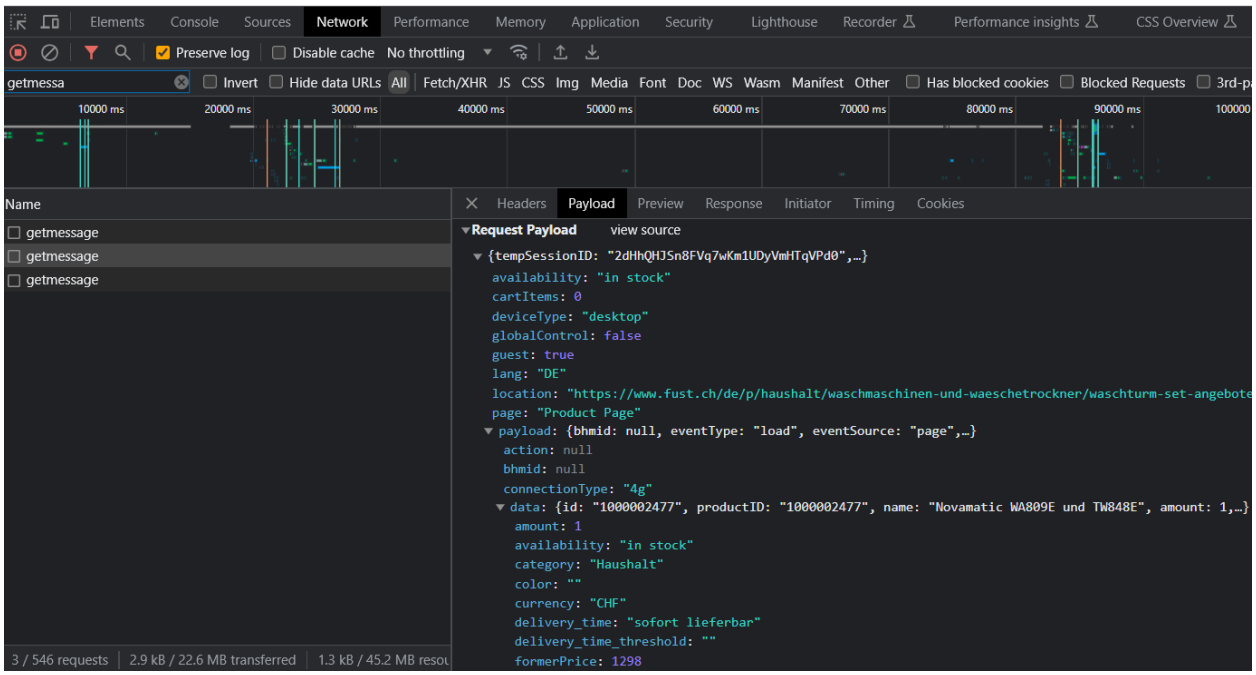

Figura 2: getMessage payload

Databaza që behamics përdorë është Redis. Redis është një sistem menaxhues me kod burimor të hapur për komunitetin, kryesisht përdorë memorien e kompjuterit për ruajtjen e të dhënave kjo ofron qasje të shpejtë dhe performancë të lartë.

Në veqanti, Redis është i përshtatshëm për aplikacione me nevoje për qasje të shpejtë në të dhënat, si ndërveprimi me sesione te përdorueseve, cashing etj.

Redis mundëson ruajtjen e të dhënave në një gamë të gjerë të strukturave te ndryshme të të dhënave si stringje(Strings), lista(Lists), sete(Sets), sete te renditura(Sorted Sets), hesh-ave(Hashes), Bitfieldste, HyperLogLog etj, për dallim nga sistemet menagjuese të të dhënave tjera si: MySQL, PostgreSQL, SQLlite të cilat janë të limituara.

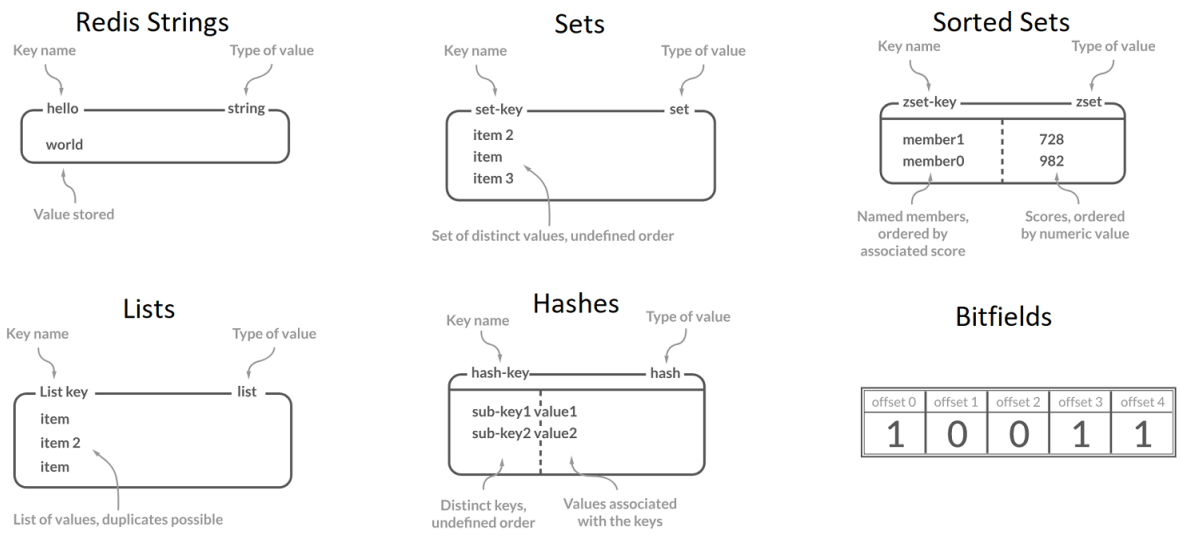

Figura 3: Strukturat e të dhënave të REDIS

2. Metodat

Funksionaliteti dhe shtjellimi i koncepteve për behamics.js është i bazuar në kodimin me anë të gjuhës programuese Javascript, përkatësisht teknikës së njohur web-scraping.

3.Përshkrimi i Aktiviteteve Kryesore

3.1 Hyrja dhe Motivimi

Çdo ditë e më shumë teknologjia e informacionit po avancohet, shërbimet fizike po shkojnë drejtë digjitalizimit si psh: vlerësimi i studenteve nga profesori përmes ueb sajtit të institucionit arsimor apo lëshimit të vërtetimeve studentore në formë elektronike me vulë elektronike nga institucioni arsimor, duke bërë kështu ndërveprimin fizik më të pakët.

Kjo e bënë të domosdoshme që ne si individ të adaptohemi me këto ndryshime dhe të avancohemi qoftë në fushën e programimit apo ndonjë fushe tjetër. Nevoja që një biznes i vogël të mbijetojë kundrejt një biznesi më të madh, është me e madhe se kurrë, kjo mund të arrihet përmes metodave të ndryshme qoftë marketingu i shërbimeve qe biznesi ofron apo ueb sajti i tyre. Dyqanet online janë shembulli më i mirë se si një biznes mund të arrijë të zhvillohet edhe me tej dhe jo të mbes vetëm si objekt fizik ku të ardhurat e tyre varen vetëm nga ardhja fizike e konsumatorëve. Pse ndodh ky fenomen?

Average Screen Time Overview

According to the latest available data, the average person spends **6 hours and 58 minutes** per day on screens connected to the internet.

That's a **1% (4 minutes)** increase over 2021's figure - equating to 5 billion+ days when extrapolated across all global internet users.

Overall, daily screen time is up **13% (49 minutes)** since 2013.

Overall, daily screen time is up **13% (49 minutes)** since 2013.

Here's a breakdown of the average internet-connected screen time since 2013:

Year	Average Screen Time	Change Over Previous Year
Q3 2013	6 hours 9 minutes	-
Q3 2014	6 hours 23 minutes	↑ 3.8%
Q3 2015	6 hours 20 minutes	↓ 0.8%
Q3 2016	6 hours 29 minutes	↑ 2.4%
Q3 2017	6 hours 46 minutes	↑ 4.4%
Q3 2018	6 hours 48 minutes	↑ 0.5%
Q3 2019	6 hours 38 minutes	↓ 2.5%
Q3 2020	6 hours 54 minutes	↑ 4%
Q3 2021	6 hours 58 minutes	↑ 1%

Figura 4: Average internet-connected screen time since 2013

Vërejmë që pothuajse ¼ e ditës shumica nga ne kalojmë kohën në telefon apo pajisje elektrike tjetër, e kështu edhe shërbimet që na nevojiten mundohemi t'i realizojmë përmes tyre. Dyqanet onlajn janë ndër më të përdorurat për këtë qëllim, por jo çdo dyqan onlajn ka sukses të njëjtë, kjo varet se sa cilësor është dyqani apo ueb sajti i biznesit, sa efikas, shpejt dhe lehtë i përdorshëm është.

Ndihma që unë mund të jap në përmirësimin e cilësisë se një dyqani onlajn përmes kompanis Behamics ka qenë motiv për mua. Nxënia e njohurive të marrura gjatë studimeve kanë qenë më se të nevojshme për ta realizuar këtë.

3.2 Përshkrimi i problemit

Në qendër të angazhimit qëndron implementimi i skriptës të mbledhjes së të dhënave, e cila ofron mbledhjen e të dhënave të një dyqani onlajn, pavarësisht nga teknologjitë e përdorur gjatë zhvillimit të saj, çdo ueb shfletues major si: Google Chrome, Microsoft Edge, Sagari, Mozilla Firefox, Opera ka një motor të Javascript-ës të integruar më vete që ekzekuton kodin në pajisjen e përdoruesit.
Krijimi i metodave të mbledhjes së të dhënave statike do jetë me i thjeshtë në krahasim me mbledhjen e te dhënave dinamike.

Sfidë e veçantë do jetë përditësimi i të dhënave në rast të navigimit të përdoruesit brenda dyqanit, të filtrimit të produkteve përmes inputeve, filtrimi të tyre përmes scrolling si dhe shqyrtimi i ndryshimeve të DOM-it në anën responsive.

Është e rëndësishme paraprakisht që të kuptohen metodat e skriptës behamics.js, qëllimi i tyre, përse përdoren, cilat metoda nuk ka nevojë te përdoren, cilat duhet te fshihen apo të shtohen. Të dhënat duhet të jenë të sakta dhe të plota ose skripta vetëm vazhdon të mbledh të dhëna gabim apo të dhëna mangu dhe kështu nuk mundemi të arrijmë qëllimin tonë.

Skripta mbledh të dhëna në këto kategori: Faqja fillestare(Home Page), Lista e produkteve(Product List), Faqja e produktit(Product Page), Lista e dëshirave(Wishlist), Karta(Cart), Blerja dhe konfirmimi(Order and Confirmation) dhe ne kategori te tjera varësisht nga nevoja e të mbledhurit ë të dhënave.

3.2.1 Skripta për mbledhjen e të dhënave për Gjirafa50

3.2.1.1 Lokacioni aktual i përdoruesit në dyqan.

Metoda _findPageType shërben për të kontrolluar se ku është përdoruesi aktualisht duke vizituar, më poshtë eshte realizuar dhe përgjigjjen mundemi me pa përmes objektit getmessage në network.

```
_findPageType: function () {
  try {
    if (document.querySelector('.html-home-page')) {
      console.log('Home Page');
      return 'Home Page';
    } else if (document.querySelector('.html-category-page')) {
      console.log('Product List');
      return 'Product List';
    } else if (document.querySelector('.html-product-details-page')) {
      console.log('Product Page');
      return 'Product Page';
    } else if (document.querySelector('.html-shopping-cart-page')) {
      console.log('Cart');
      return 'Cart';
    } else if (document.querySelector('.html-wishlist-page')) {
      console.log('Wishlist');
      return 'Wishlist';
    } else {
      return 'Other';
    }
    return document.location.pathname;
  } catch (e) {
    behamics._reportErrorToServer(e);
  }
},
```

Figura 5: Metoda _findPageType()

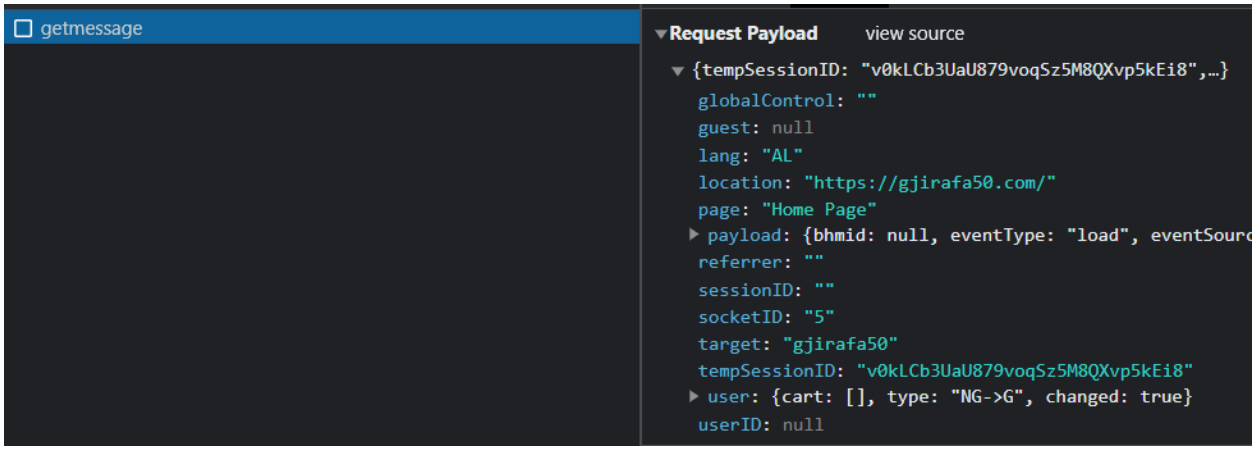

Figura 6: Rezultati i metodës _findPageType()

3.2.1.2 Mbledhja e id-ve të produkteve nga lista e produkteve.

Metoda _getListOfProductsIDs, shërben për të mbledhur id-të unike të produkteve përmes javascript selektorëve, por që preferohet selektorët të jenë opsioni i fundit për mbledhjen e të dhënave, në qoftë se ueb sajti apo klienti na ofron këto të dhëna përmes dataLayer apo ndonjë javascript objekti janë më të sigurta.

```
612       _getListOfProductsIDs: function (obj) {
613         try {
614           if (obj == 2 && behamics.loadPageSent == false) {
615             return null;
616           }
617           var tmpProductListIDArrays = [];
618           var newIDsToBeResolved = [];
619
620           var tmpProductList = document.querySelectorAll(
621             '.product-grid .product-item'
622           );
623
624           if (tmpProductList.length == 0) {
625             return 'wait';
626           }
627           for (i = 0; i < tmpProductList.length; i++) {
628             var tmpID = tmpProductList[i].getAttribute('data-productid');
629             if (
630               ProductList['resolvedIDs'].indexOf(tmpID) == -1 &&
631               typeof tmpID != 'undefined'
632             ) {
633               if (behamics._onScreen(tmpProductList[i])) {
634                 ProductList['resolvedIDs'].push(tmpID);
635                 newIDsToBeResolved.push(tmpID);
636               }
637             }
638
639             tmpProductListIDArrays.push(tmpID);
640           }
```

Figura 7: Metoda _getListOfProductsIDs()

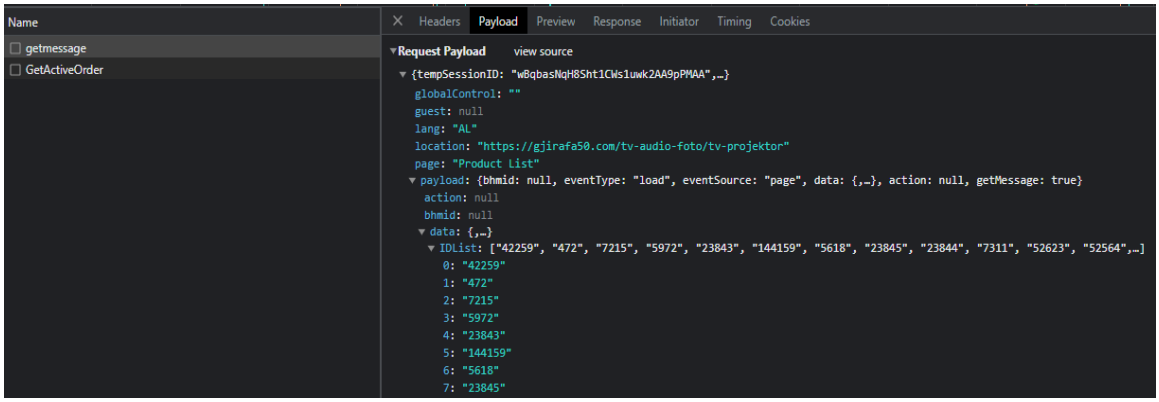

Figura 8: Rezultati i metodës _getListOfProductIDs()

3.2.1.3 Mbledhja e të dhënave të produktit nga faqja e produktit.

Metoda _getProduct() shërben për mbledhjen e të dhënave të produktit, të dhënat siç e përmendem me lartë është me e sigurt të sigurohen nga klienti përmes ndonjë objekti por si opsion të fundit kemi përdorimin e javascript selektoreve ku të dhënat i marrim nga vet DOM-i i ueb sajtit.

```
    behamics._isDataValid(payload.data);
    behamics._sendRequestToServer(behamics.variables.page, payload);
  },

  _getProduct: function (response) {
    try {
      var tempob = null;
      let amount = 1;
      let color = '';
      let price;
      let formerPrice;
      let size = '1';
      let name;
      let url = window.location.href;
      // let category = dataLayer.find((obj) => obj.ecommerce).ecommerce.detail.products[0].category;
      let category = '';
      let tempoSizeObj = {};
      let productID;
      let id;
      let rating = 0;
      let reviews = 0;
      let imgUrl = '';
      productID = document
        .querySelector('[data-productid]')
        ?.getAttribute('data-productid');
      id = productID;
      name = document.querySelector('.product-name');
      if (
        document.querySelectorAll(
          `.prices .price-value-${document
            .querySelector('[data-productid]')
            .getAttribute('data-productid')}`
        ).length > 1
      ) {
        formerPrice = document
          .querySelectorAll(
            `.prices .price-value-${document
              .querySelector('[data-productid]')
              .getAttribute('data-productid')}`
          )[0]
          .innerText.split('€')[0]
          .trim();
```

Figura 9: Metoda _getProduct

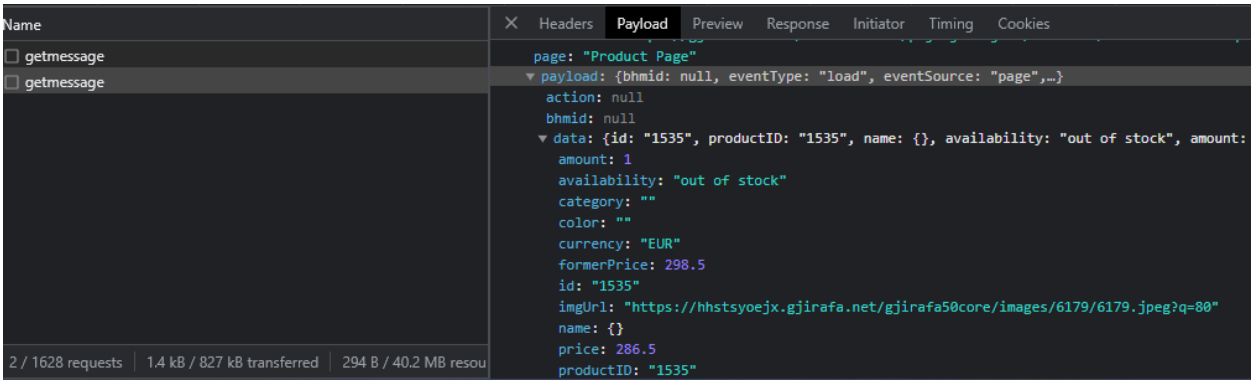

Figura 10: Rezultati i metodës _getProduct()

3.2.1.4 Përditësimi i të dhënave të reja përgjatë navigimit të konsumatorit brenda ueb sajtit.

Metoda _navigationInterceptor() është ndër metodat me të rëndësishme pse kështu? Qdo dyqan dallon për nga zhvillimi dhe teknologjitë e përdorura, disa dyqane implementojnë rifreskimin e faqes, kur klienti navigon/leviz nga një listë e produkteve në listën tjetrën, ndërsa disa dyqane tjera nuk e rifreskojnë faqen por vetëm URL-in e faqes, kështu skripta mbledh të dhëna gabim duke mos i përditësuar ato, ku të dhënat e mbledhura për listën fillestare do të jenë të njëjta edhe për listat e tjera.

Si të rregullojm këtë? Mundemi në disa mënyra varësisht se me qfarë gjendje jemi tu u merr, më poshtë kemi përdorur teknikën e interceptimit të thirrjeve *xhr* brenda ueb-it, kështu gjatë navigimit sapo të behet një thirrje xhr krahasojmë URL-in e vjetër me te URL-in e ri në qoftë se ato përputhen atëherë kjo nënkupton që përdoruesi nuk ka naviguar në faqe tjetër, por nëse ka ndodhur atëherë thërrasim metodën run() e cila i riekzekuton metodat e nevojshme për përditësim të të dhënave.

```
234   _navigationInterceptor() {
235     let oldXHROpen = window.XMLHttpRequest.prototype.open;
236     window.XMLHttpRequest.prototype.open = function (method, url, async, user, password) {
237       this.addEventListener("load", function () {
238         if (behamics.variables.page == "Product List") {
239           if (this.responseURL.split("?cid=")[1] != window.location.href.split("?cid=")[1]) {
240             try {
241               behamics.oldUrl = window.location.href;
242               behamics.run();
243             } catch (e) {
244               behamics._reportErrorToServer(e);
245             }
246           }
247         }
248       });
249
250       return oldXHROpen.apply(this, arguments);
251     };
252   },
```

Figura 11: Metoda _navigationInterceptor()

3.2.1.5 Dëgjimi për mungesën e fotove dhe dështimeve në përgjithësi.

Njëra ndër arsyet se përse ndodh fenomeni “cart abandoment” është mungesa e fotos së produktit, kjo është një minus shumë i madhë për dyqanin ngase konsumatori nuk dijë se çfarë është duke blerë, kështu ai/ajo vendos të largohet nga dyqani, kjo po ashtu ndikon në cilësinë e dyqanit sepse konsumatori mund të jap vlerësime jo të mira për dyqanin në fjalë e këto vlerësime mund t’i shohin edhe konsumatorë të tjerë.

Metoda _traceErrors() shërben për të mbledhur të dhënat në qoftë se një event gabimi ndodh nga një konsumator, ku këtu përfshihet edhe dëgjimi i dështimeve të fotografive të realizuar përmes *javascript-event-listeners*, duke marrë kështu të dhënat e nevojshme ne mund të tregojmë se cili produkt ekzakt ka dështuar dhe të këtë klienti mund të shoh dhe të marr përsipër rregullimin e tij.

```
_traceErrors() {
  try {
    window.addEventListener("error", function (event) {
      try {
        let stacktrace = event.stack;
        if (!stacktrace && event.error) {
          stacktrace = event.error.stack;
        }
        if (event.returnValue == true) {
          if (!stacktrace) {
            stacktrace = event.message;
          }
          if (JSON.parse(localStorage.getItem("tempSession"))) {
            let tmpObj = {
              type: "error",
              data: { error_message: stacktrace },
              tempSessionID: JSON.parse(localStorage.getItem("tempSession"))["behtempid"],
              target: behamics.variables.target,
              socketID: parseInt(behamics.socketID),
              page: behamics.variables.page,
              location: window.location.href,
            };
            if (navigator && navigator.sendBeacon) {
              let sendBeacon = navigator.sendBeacon(behamics.variables.behamicsApiUrl + "/api/analytics", JSON.stringify(tmpObj));
            }
          }
        }
      } catch (e) {
        behamics._reportErrorToServer(e);
      }
    });
```

Figura 12: Metoda _traceErrors()

```
window.addEventListener(
  "error",
  function (event) {
    try {
      if (navigator && navigator.sendBeacon && event.target instanceof HTMLImageElement && event.target.src.includes("cdn.shopify.com/s/files")) {
        let productID = "";
        let variantID = "";
        let page = behamics._findPageType();

        if (page == "Cart") {
          variantID = event.target.closest(".cart__card.container").getAttribute("data-variant-id");
          productID = behamics.user.cart.find((obj) => obj.id == variantID).productID;
        } else if (page == "Product Page") {
          if (event.target.closest(".product-gallery__image")) {
            productID = event.target.closest(".product-gallery__image").getAttribute("data-product-id");
          } else if (event.target.closest(".flickity-slider")) {
            productID = behamics.boxes["Product Page"]._getProduct().productID;
          }
        } else if (page == "Product List") {
          productID = event.target.closest(".product__grid-item").querySelector("div[data-id]").getAttribute("data-id");
        }
        if (JSON.parse(localStorage.getItem("tempSession"))) {
          let tmpObj = {
            type: "image_error",
            tempSessionID: JSON.parse(localStorage.getItem("tempSession"))["behtempid"],
            target: behamics.variables.target,
            socketID: parseInt(JSON.parse(localStorage.getItem("tempSession"))["behpageid"]),
            productID: productID,
            page: page,
            location: window.location.href,
          };

          let sendBeacon = navigator.sendBeacon(behamics.variables.behamicsApiUrl + "/api/analytics", JSON.stringify(tmpObj));
        }
      }
    } catch (e) {
      behamics._reportErrorToServer(e);
    }
```

Figura 13: Dëgjimi i fotografive të dështuara

Këtu shtjelluam vetëm disa nga metodat që janë të implementuar në skriptën e Behamics, duke treguar kështu idenë e teknikës *Web-Scraping* me anë të gjuhës programuese Javascript.

3.4 Platformat e Monitorimit dhe Menaxhimit të Projektit

Përgjatë praktikë profesionale, komunikimi dhe monitorimi i progresit është kryer duke shfrytëzuar platforma të ndryshme. Procesi i ndjekur për zhvillimin softuerik është bazuar krejtësisht në dy aplikime te principeve agjile. Nga Srum është shfrytëzuar teknika *WSFJ* ku tasqet janë prioritizuar në bazë të kohës, ndërsa vizualizimi i punës është bërë përmes platformës *Trello* i cili është i bazuar në Sistemin *Kanban*.

Komunikimi, mbështetja me materiale të nevojshme, këshilla kryesisht është bërë përmes platformës *Slack*. Përpos tjerash ndër me e rëndësishmja është që zhvillimi i projektit të arrihet duke mos pasur konflikte me kolegët e punës, të cilët janë duke punuar në të njëjtin projekt. Këtë e kemi realizuar përmes bashkëdyzimit të platformave *Github* dhe *Jenkins*, kështu secili nga ne do këtë versionin e projektit të përditësuar.
Jenkins shërben po ashtu për të automatizuar proceset e ndërtimit, testimit dhe implementimit të projekteve.

4. Përfundimet

Në përgjithësi ky punim i përmbledh aktivitet dhe angazhimin kryesor gjatë kryerjes së praktikës profesionale. Mes të tjerash u përmendën gjuha programuese Js, Jenkins, Trello, Web Scraping, Github, Slack. Vlen të theksohet se përveç përvojës dhe njohurisë teknike të fituar, kam fituar po ashtu edhe aftësi të buta si komunikim dhe bashkëpunim me anëtaret e ekipit më të mirë.

Përfundimisht puna praktike ishte një eksperiencë e shkëlqyer për mua, për shkak se
klientët ishin mbi 90% ndërkombëtar, me vizita të konsumatorëve në shifra milionëshe brenda muajit gjë qe e bënte sfiduese që të mos lëshohen gabime, kam arrit të shoh teknologji të ndryshme të aplikuar në dyqanet e tyre, të kuptoja intuitën e tyre e më e mira ishte se shumicën e njohurive rreth gjuhës programuese javascript dhe gjuhë të tjera e po ashtu përdorimit të platformave si Trello dhe Github i kam nxën gjatë studimeve në fakultet kjo më ka ndihmuar jashtmase.

Akronimet dhe Terminologjia

Dyqan	–	Website Online (CMS)
DOM	–	Document Object Model
Ueb sajt	–	Faqe në Internet
WSFJ	–	wighted-shortest-job-first
Shoping Cart Abandonments	–	Braktisja e kartës gjatë blerjes

Referencat

Javascript Turtorials – https://javascript.info/

Average Screen Time Statistics 2023 – https://www.crossrivertherapy.com/research/screen-time-statistics#:~:text=Average%20Screen%20Time%20Overview,screens%20connected%20to%20the%20internet.&text=The%20average%20person%20spends%20up,hours%20looking%20at%20a%20screen