ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Bakalářská práce

Vizualizace aktuálního stavu výrobní linky

Natálie Kaslová
Otevřená informatika (BS) - Počítačové hry a grafika

Květen 2024
https://gitlab.fel.cvut.cz/kaslonat/monitoring_bp
Vedoucí práce: Ing. David Sedláček, Ph.D.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kaslová** Jméno: **Natálie** Osobní číslo: **499177**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Specializace: **Počítačové hry a grafika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Vizualizace aktuálního stavu výrobní linky

Název bakalářské práce anglicky:

Visualization of the current state of the production line

Pokyny pro vypracování:

1) Seznamte se s datovým modelem výrobní linky pro výrobu klimatizačních jednotek.
2) Dle fyzické dispozice linky a nastudovaném datovém modelu navrhněte vizualizaci dynamického chování linky - pohyb jednotlivých produktů na lince, přidávání a odebírání produktů v klíčových místech, metainformace, chybové a normální stavy. Dílčí části návrhu konzultujte s odpovědnou osobou.
3) Navrhněte vhodné webové technologie pro implementaci vizualizace a implementujte ji. Při návrhu předpokládejte možná budoucí rozšíření a přeuspořádání linky.
4) Postupujte dle metodiky UCD, finální vizualizaci otestujte s cílovou skupinou uživatelů (pokud je dostupný malý vzorek uživatelů v cílové skupině, vyberte vhodnou náhradu po konzultaci s vedoucím práce minimálně do celkového počtu pěti uživatelů).

Seznam doporučené literatury:

1] Components, V. Visual Components: 3D Manufacturing Simulation and Visualization Software—Design the Factories of the Future. Available online: https://www.visualcomponents.com (accessed on 10 April 2020).
2] T. Lowdermilk, User-Centered Design, O'Reilly Media, 2013

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. David Sedláček, Ph.D. katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **16.02.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

Ing. David Sedláček, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studentky

Poděkování / Prohlášení

Ráda bych poděkovala svému vedoucímu práce, Ing. Davidu Sedláčkovi, Ph.D., za vedení, ochotu a odbornou pomoc s vypracováním této práce.

Dále děkuji společnosti Daikin za možnost vytvoření této práce, zejména panu Václavu Bauerovi za jeho trpělivost, vstřícný přístup a veškerý čas, který mi věnoval.

A nakonec děkuji své rodině, která mě za celou dobu studia podporovala.

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

.....

V Praze dne 24. 5. 2024
Natálie Kaslová

Abstrakt / Abstract

Bakalářská práce se zabývá vytvořením webové aplikace, která bude vizualizovat aktuální stav výrobní linky ve společnosti Daikin. Je zde představena logiky linky a proces výroby klimatizačních jednotek, které jsou potřeba pro pochopení problematiky. Následuje popis návrhu, kde jsou vysvětleny změny provedené na modelu linky, zejména nově používané druhy pozic a strukturální změny na lince, a nakonec samotná implementace výsledné aplikace. Ta využívá ASP.NET MVC architekturu, komunikace mezi serverem a klientem je řešena pomocí SignalR knihovny a o samotné grafické zobrazení vizualizace se stará knihovna Three.js.

Klíčová slova: webová aplikace, vizualizace, výrobní linka, ASP.NET, MVC, Three.js, SignalR

The bachelor thesis focuses on creating a web application that will visualize the current state of the production line at Daikin company. It presents the logic of the line and the manufacturing process of air conditioning units, which is neccessary for understanding the topic. This is followed by a description of the design, where the changes made to the line model are explained, particularly the newly used types of positions and structural changes to the line, and finally, the implementation of the resulting application. The application utilizes the ASP.NET MVC architecture, communication between the server and client is handled by SignalR library, and the graphical visualization is managed by the Three.js library.

Keywords: web application, visualization, assembly line, production line, ASP.NET, MVC, Three.js, SignalR

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

Společnost Daikin [1] je jedním z nejprestižnějších výrobců klimatizačních jednotek. Jedná se o japonskou značku, která má svá sídla a výrobní závody po celém světě. Protože proces výroby na výrobní lince je složitý a může během něho vzniknout mnoho různých poruch ať už na straně operátora linky, softwaru nebo hardwaru, je potřeba mít nástroje, pomocí kterých se rychle opraví a linka se vrátí co nejrychleji zpět do provozu.

Cílem této práce je pro jeden takový závod sídlící v Plzni vytvořit webový systém, který bude v reálném čase zpracovávat data z výrobních linek a vytvoří z nich podrobný model výroby. Aplikace by měla značně zefektivnit a zpřehlednit práci s daty, protože bude vizualizovat databázi a díky tomu se zrychlí řešení problémů.

Kapitola 2
Analýza problému

V rámci této kapitoly bude přiblížena problematika. Nejdříve se rozeberou požadavky, které má aplikace splňovat. Dále následuje rozbor výrobní linky, který je potřeba pro pochopení funkcionality a na který navazuje analýza modelu výrobní linky, od kterého se později bude odvíjet řešení práce. V poslední řadě budou zmíněny podobné, již existující programy vizualizací.

2.1 Systémové požadavky

V rámci zadání projektu byly stanoveny systémové požadavky, aby aplikace splnila očekávání a umožnila DICZ (Daikin Industries Czech Republic s.r.o.) snadné nasazení do provozu.

2.1.1 Funkční požadavky

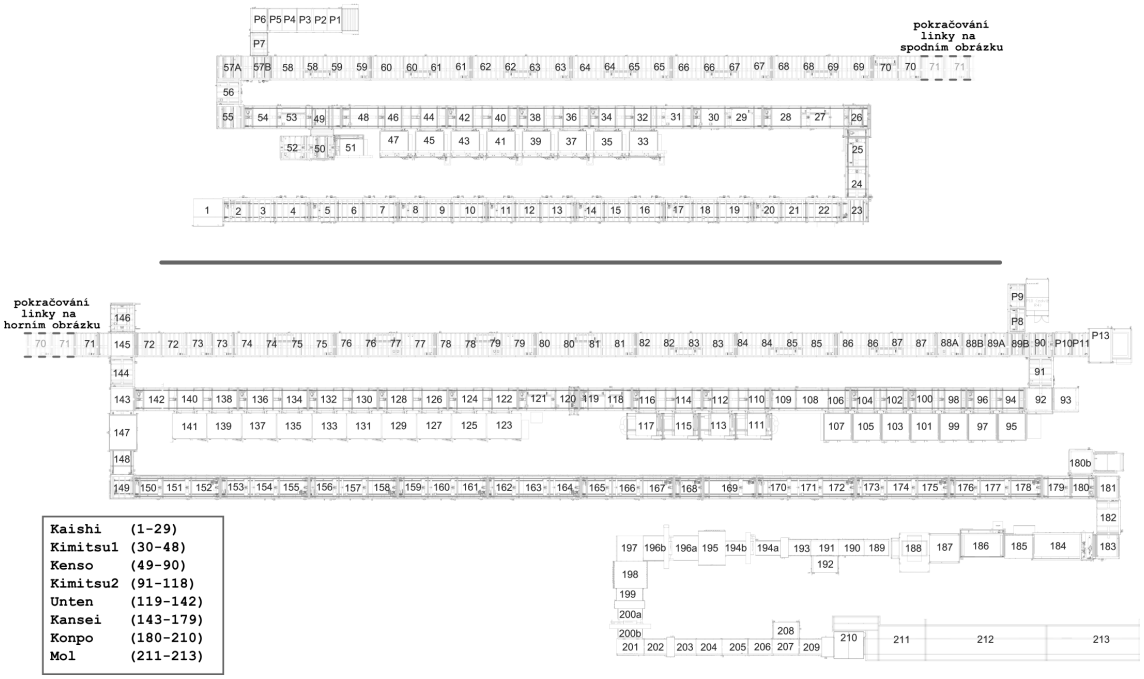
- Živé zobrazení dat
- Vizualizace výrobní linky R5
  - Zobrazení rozložení pozic
  - Pohybující se jednotky na lince
- Výpis jednotek nacházejících se v opravě
- Dohledávání informací k jednotkám

2.1.2 Technické požadavky

- Webová aplikace
- MVC architektura
- Práce s databází

2.2 Popis výrobní linky

Ve výrobním závodě Daikinu se sídlem v Plzni (DICZ) se aktuálně (ke dni 4.5.2024) nachází 9 výrobních linek, které se zabývají výrobou klimatizačních jednotek. Pro nejnovější a největší linku s označením R5 bude v rámci projektu vytvořena webová aplikace zobrazující aktuální stav výrobní linky. Pro představu rozsahu linky R5 je na obrázku 2.1. vidět její rozložení a na obrázku 2.2. pohled na skutečnou linku.



Obrázek 2.1. Rozložení linky R5 rozdělené na dva úseky. Pozice jsou označeny čísly. Výroba začíná na pozici číslo 1 a končí na pozici číslo 213.

Obrázek 2.2. Pohled na celou linku R5. Nejbližší roh je pozice číslo 183.

Protože se jedná o japonskou společnost, mnoho používaných názvů je v původním znění. Jejich význam je vysvětlen v příloze A - Zkratky a slovník.

Linka je složena z několika částí, kde se v každé části provádí kus z procesu výroby jednotky. Části se dále dělí na jednotlivé pozice (u linky R5 jich je přes 200), které jsou označeny čísly.

Části a jejich rozložení je následující:

- *Kaishi* (pozice 1-30)
- *Kimitsu*1 (pozice 31-48)
- *Kenso* (pozice 49-90)
- *Kimitsu*2 (pozice 91-118)
- *Unten* (pozice 119-142)
- *Kansei* (pozice 143-179)
- *Konpo* (pozice 180-210)
- *Mol* (pozice 211-213)

U pozic v částech *Kaishi*, *Kenso*, *Kansei* a *Konpo* stojí operátoři linky, kteří na jednotce provádějí příslušné úkony, nebo se na jednotce provádějí automatizované úkony, např. pájení. *Kimitsu*1, *Kimitsu*2 a *Unten* jsou testovací stanice. Jednotka je odkloněna z linky a zajede do testovací stanice, kde jsou prováděny automatizovaného procesy. Ve stanici jednotka zůstává, dokud není proces dokončen.

První pozice v každé části je vyhodnocovací pozice. Tou je buď křižovatka (pozice, na které je možné jednotku odklonit z výroby), nebo kontrola kvality, kde má operátor za úkol zkontrolovat správnost provedených úkonů. Pokud jednotka nevyhovuje, je označena jako vadná (NG) a nemůže pokračovat do další části, musí se nejdříve opravit.

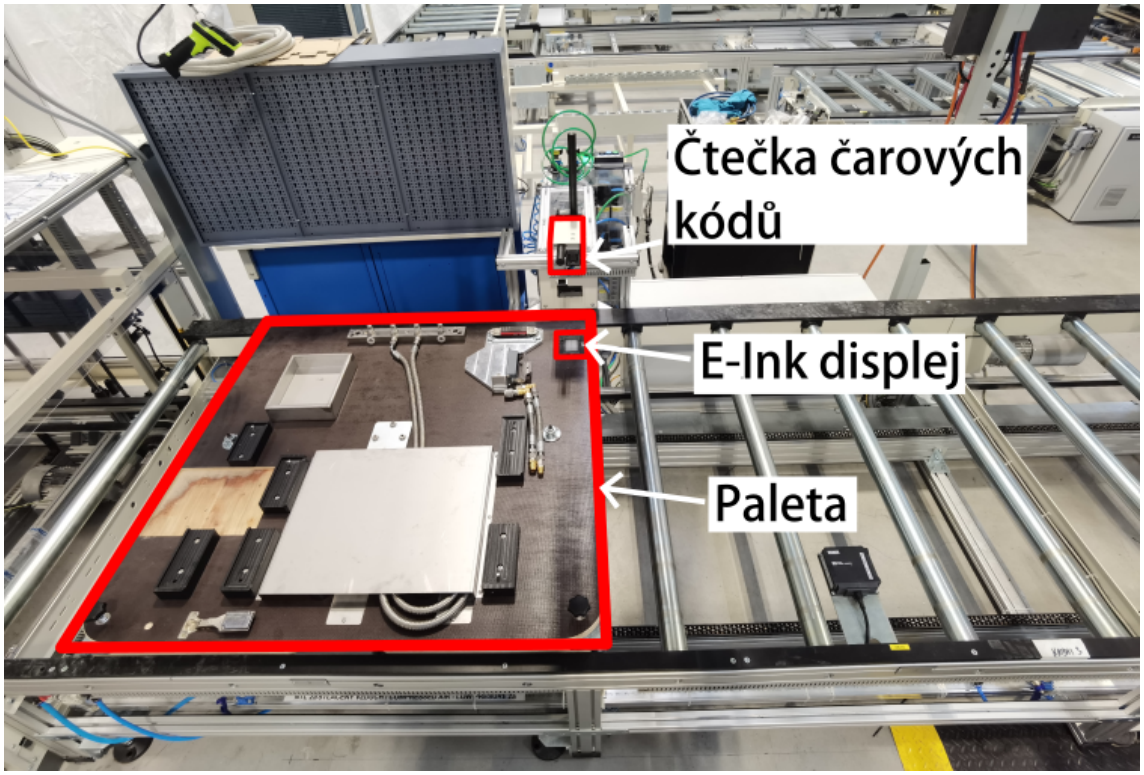
Jednotka na pozici stráví průměrně 50 sekund a celý proces její výroby trvá zhruba 2,5 hodiny.

2.3 Proces výroby jednotky

Každá jednotka začíná proces své výroby přiřazením k paletě. Paleta umožňuje jednotce pohyb po pozicích, nese informace o jednotce a díky čipům umožňuje komunikaci s ostatními komponentami linky.

2.3.1 Kaishi

Na pozici číslo 1, v části *Kaishi*, dochází k vytvoření a nahrání dat o jednotce do RFID čipu, který se nachází na paletě. Zároveň se data nahrají do E-Ink displeje na paletě, který umožňuje vizuální identifikaci jednotky během výroby. Následně se ověří správnost dat v E-Ink displeji pomocí čtečky čárových kódů, která se nachází na pozici. Paleta, E-Ink displej i čtečka čárových kódů jsou vyznačeny na obrázku 2.3. Na závěr se vygenerují data a vloží do tabulky *KumiSkkr* (tabulka obsahující seznam používaných palet) a data o jednotce se vloží do tabulky *PartKaishi* (tabulka obsahující všechny jednotky v *Kaishi*).

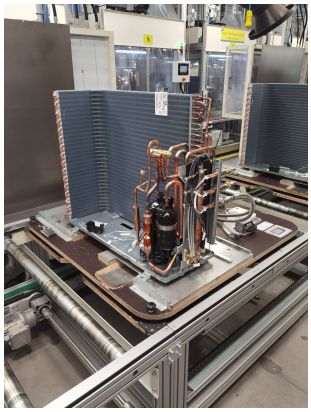


Obrázek 2.3. Pohled na pozici číslo 1 se zvýrazněnou paletou, E-Ink displejem a čtečkou čárových kódů

V části *Kaishi* (obrázek 2.4) začíná samotná výroba jednotky. Začíná nasazením přiřazené spodní kostry na paletu (kostra se připravuje na separovaném pracovišti) a pokračuje nasazením kompresoru a výměníku tepla (jednotka s kompresorem je na obrázku 2.5 a jednotka s výměníkem tepla na obrázku 2.6). Zároveň na několika pozicích dochází k dusíkování a pájení. Před každou pájecí pozicí se nachází jedna dusíkovací, která umožňuje lepší výsledky během následného pájení (méně škodlivin a větší efektivita).

Obrázek 2.4. Pohled na část *Kaishi*

Obrázek 2.5. Jednotka po nasazení kompresoru



Obrázek 2.6. Jednotka po nasazení výměníku tepla

2.3.2 Kimitsu1

Následuje část *Kimitsu*1 (obrázek 2.7), kde se nacházejí testovací stanice (pozice 33, 35, 37, 39, 41, 43, 45 a 47). Nejprve se jednotka vyvakuuje a následně se naplní směsí plynů, které se snadno detekují. Operátor pak použije sondy, které dokáží únik plynu

rozpoznat. Pokud operátor žádnou chybu nenajde a zároveň je tlak v jednotce stabilní, zapíše se do RFID čipu výsledek testování OK. V opačném případě se zapíše NG.

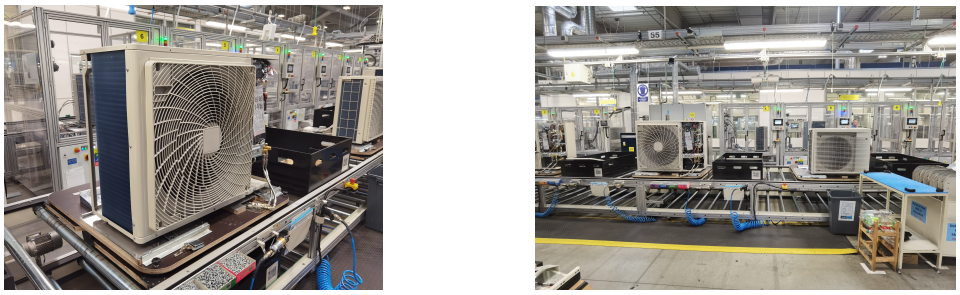
Obrázek 2.7. Pohled na testovací stanice v části *Kimitsu*1

2.3.3 Kenso

Na pozici 49, která je křižovatkou, se kontroluje výsledek testování. V případě, že výsledek je OK, jednotka pokračuje dál přes pozice 53 a 54 v procesu výroby. Pokud však výsledek byl NG, jednotka je odkloněna na pozice 50 a 52 a je potřeba provést její opravu. Po dokončené opravě jednotka jede do testovací stanice pro opravené jednotky na pozici číslo 51 a až po hotovém testu může odjet zpět na pozici 49 a pokračovat ve výrobě.

Dále se na části *Kenso* se k jednotce přiřadí Parts Box (box s díly, viz obrázek 2.8). Na křižovatce číslo 49 se načtou data z RFID čipu a podle nich se rozhodne, jestli k jednotce patří nějaký Parts Box. Pokud ano, tak operátor musí vložit příslušný Parts Box na pozici P1 a následně ho načíst čtečkou, čímž potvrdí jeho správnost. Parts Box pokračuje přes pozice P2-P7 a zařadí se za jednotku, ke které patří. Na konci části Parts Boxy zase opustí linku přes pozice P8 a P9, nebo P10, P11, P13 (podle stavu jednotky, jestlii je NG nebo OK). Jednotka projíždějící *Kenso* je vidět na obrázku 2.9. Na konci části je jednotce vygenerovaná hodnota *Kiban*.

Obrázek 2.8. Detail na Parts Box



Obrázek 2.9. Jednotka projíždějící částí *Kenso*

2.3.4 Kimitsu2

V části *Kimitsu*2 (obrázek 2.10), kde se nacházejí další testovací stanice (95, 97, 99, 101, 103, 105, 107, 111, 113, 115 a 117), se jednotka nejdříve vyvakuuje a pak dojde k naplnění jednotky chladivem. Celé plnění je velmi sofistikovaná operace, při které se kontroluje hmotnost naplněného chladiva na přesnost desetiny gramu. Celkově se na jednotku použije 1,2-4kg chladiva.

Obrázek 2.10. Vlevo je vidět část *Kimitsu*2, vpravo je konec části *Kenso*

2.3.5 Unten & Kansei

Část *Unten* je poslední část s testovacími stanicemi (123, 125, 127, 129, 131, 133, 135, 137, 139, 141). Na pozici 120 nejdříve probíhají elektrické testy, kde operátor kontroluje zkrat jednotky po zapojení do eletrického proudu. Poté se ve stanicích kontroluje správná funkcionalita všech větráků a otevíracích částí jednotky. Po úspešném otestování se do dat jednotky nahraje vygenerovaná hodnota *Kiban* (vygenerována byla na konci části *Kenso*).

V části *Kansei* probíhá dokončování jednotky. Jednotlivé matice u vstupních částí jednotky jsou dotaženy a jednotka je polepena několika štítky. Jedná se o identifikační štítek firmy, identifikační štítek jednotky a štítky určené evropskými normami.

2.3.6 Konpo & Mol

V části *Konpo* se jednotka zabalí do balícího kartonu a polepí se identifikačním štítkem (obrázek 2.11), který používá kombinaci *Kibanu* a *Poscode* (*Poscode* nahrazuje *Seban*). Jednotka pokračuje do části *Mol* odkud je odvezena do skladů.

Obrázek 2.11. Pohled na zabalenou jednotku

V části *Mol* probíhá třízení jednotek podle velikosti a druhu. Následně se přesunou do kamionů, které je odvezou do skladů.

2.4 Analýza modelu

Model bude pracovat s dvěma hlavními komponentami. První jsou pozice, z kterých je vytvořen tvar a rozložení linky. Jsou rozlišeny identifikačními čísly. Druhá komponenta je samotná jednotka, která se pohybuje po pozicích. Ke každé jednotce bude možné dohledávat informace, např. sekvenční číslo, název modelu a stav (výsledky jednotlivých pozic).

2.4.1 Jednotka

Identifikátor jednotky je kombinace *Sebanu* a *Sjuni* (případně *Sebanu* a *Kibanu*). Na základě těchto dat se dají o jednotce vyhledat informace v tabulkách. Na jedné lince se současně vyrábí několik různých modelů jednotek. Modely jsou rozlišeny *Sebanem*, *Sjuni* pak určuje sekvenční číslo výroby.

2.4.2 Pozice

Pozice určuje dráhu pohybu jednotky. V každý moment se na pozici může nacházet pouze jedna jednotka. Existují také speciální druhy pozic a těmi jsou:

- **Křižovatka** - druh pozice, která pohybuje jednotkou dvěmi směry podle jejího stavu. Klasicky pošle jednotku do další části linky. V případě, kdy je stav jednotky NG, ji přesměruje mimo linku, kde jí individuálně opraví příslušní pracovníci. V momentě, kdy je oprava dokončena, je jednotka vložena zpět na křižovatku a pokračuje ve výrobním procesu.
- **Testovací pozice** - místo, kde jednotka zůstává předem neznámý čas. Každý test pojme pouze jednu jednotku. Jednotce je přidělen první prázdný test v části, kde se aktuálně nachází.

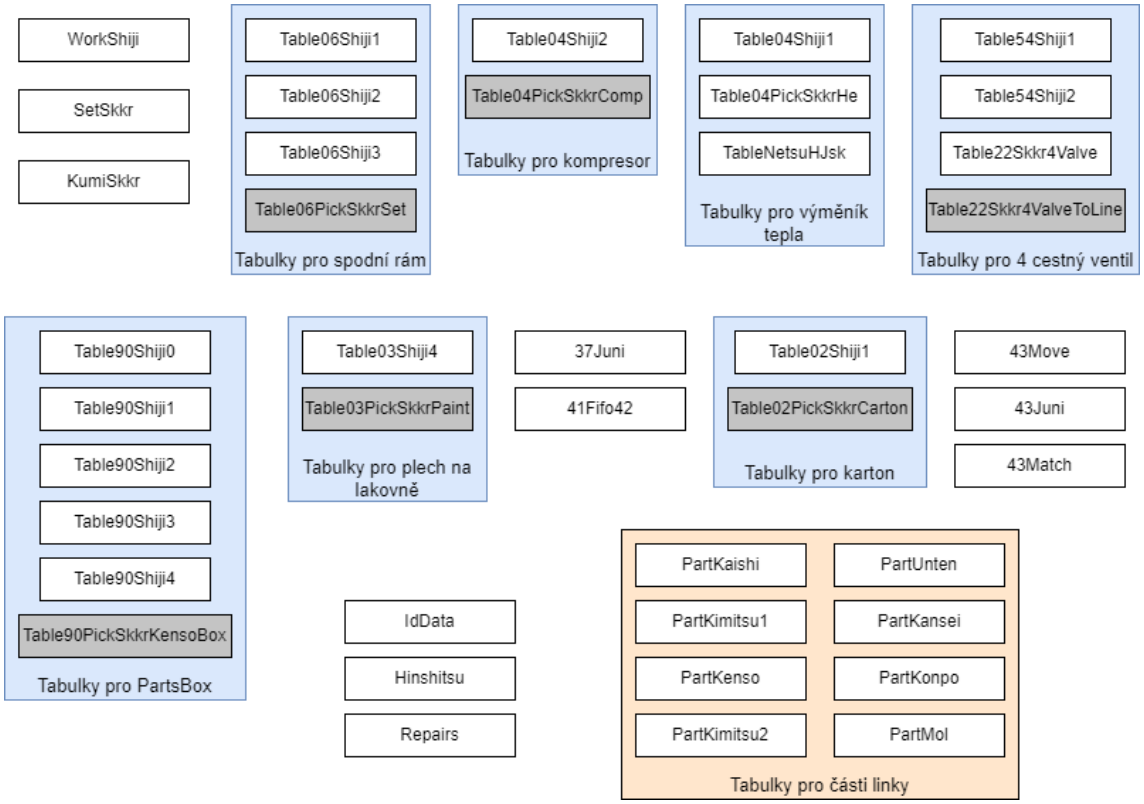
Struktura speciálních druhů pozic je znázorněna na obrázku 2.12.

Obrázek 2.12. Analýza linky R5 rozdělené na dva úseky s vyznačenými speciálními pozicemi

2.4.3 Databáze

V DICZ se používá relační databáze, ve které se nepoužívá žádné cizí klíče. Mezi jednotlivými tabulkami neexistuje žádný vztah.

V průběhu výroby jednotky je používáno přes 40 různých tabulek (jejich schéma je vidět na obrázku 2.13). Využívají se buď pro uchování dat o jednotce, pro určení umístění jednotky na lince, pro kontrolu správnosti dat nebo pro správné přiřazení dílů k jednotce.



Obrázek 2.13. Schéma databáze

Tabulky ve formátu *Table__Shiji_* (kde _ nahrazuje číslo) obsahují data sloužící pro výběr správných dílů operátorem. První číslo značí číslo pozice, na které se tabulka využívá a druhé číslo označuje číslo operátora. Tabulka ve formátu *Table__PickSkkrt__* pak obsahuje výsledky z předchozích tabulek pro výběr dílů.

Význam ostatních tabulek je následující:

- *WorkShiji* - seznam jednotek, které se mají nahrát na paletu
- *SetSkkr* - seznam spárovaných spodních rámů s kompresorem, který se používá pro přiřazení dílů k paletě
- *KumiSkkr* - seznam nahraných jednotek na paletě, který se používá ke kontrole jednotek ve výrobě
- *TableNetsuHJsk* - fronta výměníků tepla z výměníkové linky na hlavní linku (kde se vyrábí jednotky)
- *Table*54*Skkr*4*Valve* - fronta dat pro tisk na pracovišti 4 cestného ventilu
- *Table*54*Skkr*4*ValveToLine* - výsledek z pracoviště 4 cestných ventilů
- 37*Juni* - seznam výrobních štítků pro tisk (štítek nalepený na jednotce pro její identifikování)
- 41*Fifo*42 - fronta jednotek mezi pozicemi 132 a 133
- 43*Move* - fronta jednotek mezi pozicemi 134 a 136
- 43*Juni* - fronta dat pro tisk packing case štítků (štítek nalepený na kartonu pro identifikaci ve skladech)
- 43*Match* - fronta dat pro kontrolu vytištěných štítků
- *IdData*, *Hinshitsu* a *Part__* jsou vysvětleny níže

Nejobsáhlejšími tabulkami jsou *IdData* a *Hinshitsu*, ve kterých jsou uloženy všechny informace o jednotce a nastavení různých procesů. V tabulce *IdData* je celkem 175 atributů a tabulka má primární klíč kombinaci *Sebanu* a *Sjuni*. Záznam jednotky je v

tabulce od začátku její výroby a postupně se její data doplňují, případně mění. Při balení jednotky se její záznam vloží do tabulky *Hinshitsu*, která má autributů 166 a primární klíč je kombinace *Sebanu* a *Kibanu*.

Může se stát, že jednotka, která již byla zabalena, tudíž má záznam v *Hinshitsu*, bude linkou projížděť znovu, protože s ní něco nebylo v pořádku a je potřeba jí opravit, nebo protože byla vybrána na náhodné překontrolování kvality.

Pro každou část linky existuje tabulka v databázi, jejíž schéma je v tabulce 2.1. Tabulky pro jednotlivé části jsou rozlišeny názvem tabulky, který začíná *Part* a pokračuje názvem části, jedná se tedy o tabulky: *PartKaishi*, *PartKimitsu*1, *PartKenso*, *PartKimitsu*2, *PartUnten*, *PartKansei*, *PartKonpo* a *PartMol*. Do tabulky jsou vkládany nové záznamy v případě, kdy jednotka vstupuje do dané části a ve stejné transakci se smaže záznam z tabulky části, kterou opustila. Nový záznam má aktualizovanou hodnotu DATEKEY - čas vložení záznamu do tabulky ve formátu *YYYYMMDDhhmmss*[1].

Seban značí čtyřčíslí nahrazující jméno modelu. Jednotka se dá rozdělit na 3 důležité části, kde každá z nich má svůj vlastní *Seban*. *Seban*1 označuje model celé jednotky, *Seban*2 identifikuje výměník tepla a *Seban*3 identifikuje spodní kostru. *Cnt* značí počet kusů, v aktuální době nepoužívaný údaj. Použití atributů *Code*1 a *Code*2 se liší podle jednotlivých tabulek.

Part ...	
DateKey (PK)	char(14)
Kishu	char(14)
Sline	char(6)
Zuban	char(18)
Seban1	char(4)
Sjuni	char(15)
Kiban	char(8)
Seban2	char(4)
Seban3	char(4)
Cnt	char(4)
Code1	char(2)
Code2	char(2)

Tabulka 2.1. Schéma databáze pro tabulku části.

Záznamy v tabulce *PartKaishi* mohou vypadat jako v tabulce 2.2.

Stejný systém vkládání a mazání dat funguje pro tabulku jednotek v opravě (*Repairs*), pouze data mají jiný formát (viz tabulka 2.3). Záznam jednotky je do tabulky přidán v případě, kdy jednotka opouští linku a jde na opravu, a je vymazán v případě, kdy se na linku zase vrací. Hodnota *NG_Code* značí kód chyby a *Position* číslo křižovatky, na které jednotka opustila linku.

[1] Značení podle normy ISO 8601

DateKey	20240320132415	20240320134026	20240320134501
Kishu	5MXM90A2V1B9	4MXM80A2V1B9	RZAG140N2Y1B
Sline	111500	111500	111500
Zuban	00000000000000	00000000000000	00000000000000
Seban1	0803	0802	0773
Sjuni	200001011080301	200001011080201	200001011077301
Kiban	0J000081	0J999991	0J999991
Seban2	0938	0938	0953
Seban3	0000	0000	0000
Cnt	00	00	00
Code1	00	00	00
Code2	00	00	00

Tabulka 2.2. Ukázka záznamů v tabulce *PartKaishi* (záznamy jsou formátovány do sloupce)

Repairs	
DateKey (PK)	char(14)
Position	char(15)
Seban	char(4)
Sjuni	char(15)
NG_Code	char(4)

Tabulka 2.3. Schéma databáze pro jednotky v opravě

Záznamy v tabulce *Repairs* mohou vypadat následovně (tabulka 2.4):

DateKey	Position	Seban	Sjuni	NG_Code
20240320135420	147	0767	200001011076702	5272
20240320135338	49	0767	200001011076701	2048
20240320135256	91	0770	200001011077001	3709

Tabulka 2.4. Ukázka záznamů v tabulce *Repairs*

2.5 Podobné práce

Vytvoření aplikace, která řeší stejnou problematiku jako tento projekt, je složitý proces, který buď vyžaduje mnohé úpravy ve výrobním informačním systému, nebo musí být požadavky pro její provoz zohledněny při návrhu nového výrobního systému. Proto výsledná aplikace tohoto projektu nezobrazuje výrobní linku 1:1, ale zobrazuje pouze přibližný pohyb jednotek na lince.

Aplikace takového charakteru pak nebývají veřejně dostupné, ale jsou využívány pouze interně, protože pracují s citlivými daty o výrobě.

Podobnými aplikacemi, které vizualizují výrobu, jsou simulace, které však nepracují s živými daty. Následuje příklad simulačního softwaru, knihy a práce zabívající se simulací výrobní linky. Nakonec je představena práce, která porovnávala monitoring výroby v AR a webovém rozhraní.

2.5.1 Visual Components

Jedním ze simulačních softwarů je Visual Components [2], který poskytuje nástroje pro simulaci výroby širokého spektra podniků, od malých pekáren až po automobilový průmysl (ukázka simulace je vidět na obrázku 2.14). Simulace výroby vyniká svým dynamicým přístupem závislím na čase a poskytuje komplexní a realistické pochopení výrobních procesů. Nabízí vizuální a animovanou reprezentaci komplexních interakcí, umožňuje výrobcům přesně plánovat, optimalizovat a předvídat výsledky jejich výrobních procesů.

Obrázek 2.14. Simulace výrobní linky vytvořená pomocí Visual Components [3]

2.5.2 Production line 3D visualization monitoring system design based on OpenGL

Práce podobného charakteru, *Production line 3D visualization monitoring system design based on OpenGL* [4], popisuje systém pro monitorování provozu výrobní linky v reálném čase prostřednictvím 3D vizualizačního rozhraní. Klíčové komonenty systému zahrnují modul modelování výrobní linky, modulu sběru a předzpracování procesních informací a modulu vizualizace procesních informací. Modul modelování výrobní linky vytváří 3D modely jednotlivých komponent linky, jejich umístění a pohyb pomocí OpenGL. Modul sběru a předzpracování procesních informací se používá ke sběru procesních informací z výrobního prostředí v reálném čase a zajištění komunikace s klienty na straně serveru. Modul vizualizace procesních informací používají klienti k zobrazení aktuálního stavu výrobní linky a umožňuje jim sledovat operace obrobků a tok materiálu.

Systém monitorování vizualizace výrobní linky je realizován kombinací technologie modelování výrobní linky založené na OpenGL, komunikace mezi serverem a klientem a správy databáze. Cílem vizualizace výrobní linky je optimalizovat strukturu výroby, snížit výrobní náklady a poskytnout efektivní datovou podporu pro podnik s cílem dosáhnout maximálních přínosů. Ve zmíněné práci se zaměřují na reformu systému, vyvinutí výrobního plánu a nastavení parametrů maximální výrobní kapacity pro provedení simulačního experimentu výrobní kapacity. Je nutné provést simulační experiment výrobní linky až po dokončení modelování vizualizace výrobní linky. Do simulačního systému se vkládají různé parametry a výsledky jsou zaznamenány. Optimální rychlost

dopravního pásu a časový interval podávání surovin jsou následně analyzovány a určeny k maximalizaci kapacity výrobní linky.

Návrh systému monitorování a vizualizace výrobní linky ve 3D založený na OpenGL umožňuje podnikům pozorovat jak provoz výrobní linky, tak i provozní proces, který zahrnuje informace o procesu v reálném čase, historické informace o procesu a informace o selhání procesu.

2.5.3 Simulation Modeling and Arena

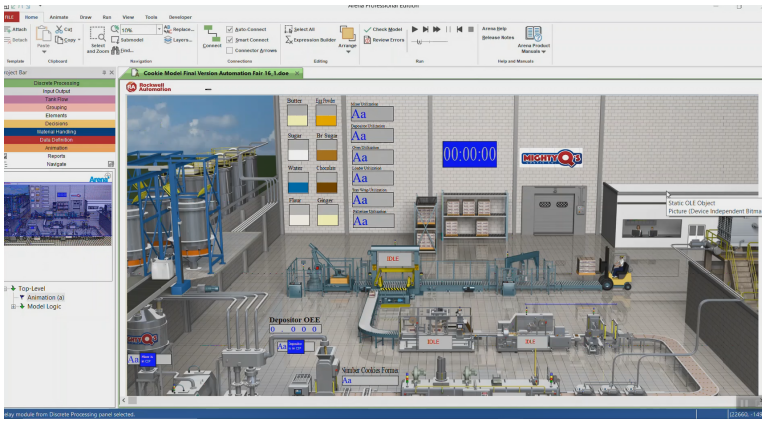
Návrh tohoto projektu se inspiroval níže popsaným návodem podle knihy *Simulation Modeling and Arena* [5]. Kniha poskytuje návod, na jehož základě by čtenáři měl poskytnout počáteční nápady, jak přistupovat k simulačnímu modelu. Dosahuje toho pomocí zodpovězení základních modelových otázek, které čtenáře provedou kroky modelování simulace, aby následně mohl vypracovat podrobné řešení problému. Otázky jsou následující:

- Co je systém? Jaká data systém poskytuje?
- Co jsou entity? Jaká data musí být nahrána nebo zaznamenána pro každou entitu? Měl by systém vědet o entitách?
- Jaké zdroje jsou entitami používány? Jaká entita používá jaký zdroj a jak?
- Jaký je datový tok? Načrtni proces nebo vytvoř diagram datového toku.

Kde definice entity je: Objekt zájmu v systému, jehož pohyb nebo operace může způsobit vznik události.

Kniha následně rozebírá, jak modelovat systém pásového dopravníku, který se používá na výrobní lince v tomto projektu.

Nakonec se kniha zaměřuje na modelování vytvořeného modelu v aplikaci Arena (ukázka aplikace na obrázku 2.15) a následné testování a analýzu výsledné simulace.



Obrázek 2.15. Ukázka Arena Simulation Software [6]

2.5.4 Pervasive Augmented Reality to support real-time data monitoring in industrial scenarios

Článek, který se zabývá použitím rozšířené reality (AR) pro podporu monitorování dat v reálném čase v průmyslových scénářích: *Pervasive Augmented Reality to support real-time data monitoring in industrial scenarios* [7] se zaměřuje na hodnocení vizualizace výrobní haly a provedení uživatelské studie, která zkoumá efektivitu a uživatelskou přívětivost AR řešení. Výsledky jsou takové, že AR technologie představuje významný přínos a může výrazně zlepšit efektivitu a přesnost výrobních procesů, zároveň poskytuje cenné poznatky pro další inovace a implementace v průmyslových prostředích.

V této práci se však omezíme pouze na webovou vizualizaci, protože jejím cílem je umožnit pracovníkům řešit problémy na lince bez nutnosti přítomnosti u linky tím, že aplikace vizualizuje databázi. Vizualizace pomocí AR také zprostředkovává vizualizaci databáze, avšak uživatel musí být u linky přítomný a používat speciální zařízení.

Kapitola 3
Návrhový vzor

V této kapitle se nejdříve rozeberou změny, které je potřeba provést pro implementaci aplikace. Jedná se o vysvětlení nově používaných druhů pozic a o strukturální změny na lince. Následně se představí technologie, které aplikace bude využívat, a na závěr se kapitola bude věnovat návrhu uživatelského rozhraní.

3.1 Úprava modelu

Hlavní logická komponenta modelu je pozice. Ta shromažďuje informaci o jednotkách, které se na ní nacházejí, a odkaz na následující pozici, které jednotku předají. Od třídy pozice (Position) dědí všechny ostatní druhy pozic, kromě jednotky.

Pro každou pozici platí, že jednotku nepošle na následují pozici, jestliže záznam jednotky se ještě neobjevil v části linky, do které patří následující pozice. V případě, že část linky, do které patří aktuální pozice, není shodná s částí linky, v které se nachází záznam jednotky (jednotka se již má nacházet v další části nebo se přesouvá do opravy), pohybuje pozice jednotkou se zvýšenou rychlostí, aby změnu co nejrychleji vyrovnala.

Křižovatka (CrosswayPosition) se využívá v případě, že jednotka je vadná a z linky se přesune na opravu, případně se na linku vrací z opravy. Jednotka k tomu využívá vedlejší pozice (vedlejší ve smyslu, že nejsou součástí hlavního směru výroby). Pro zjednodušení tento proces bude nahrazen animací mizení/objevování se jednotky přímo na pozici křižovatky a vedlejší pozice se zcela vypustí (pozice 51-53, 93, 144, 146, 180b, 192, 208). Zároveň se zcela vypustí zobrazování parts boxů a ním i pozice, na kterých se připojují k lince (P1-P7) a odjíždějí z linky (P8-P13).

Všechny vyhodnocovací pozice, tj. první pozice každé části, se budou brát jako křižovatky. Kromě pozic skutečných křižovatek využívá strukturu i první a poslední pozice celé linky. Je to kvůli stejné funkcionalitě - přidávání jednotky na linku, případně odebírání jednotky z linky.

Testovací stanice bude zastoupena dvojpozicí. První část (TestCross), která se stará o přesměrování jednotky do druhé části, samotného testu (TestPosition), pokud byl jednotce přiřazen. Tam jednotka zůstává do té doby, dokud se její záznam neobjeví v další části linky.

K úseku testů patří i první pozice testů (FirstTestPosition) a poslední pozice testů (LastTestPosition), které se starají o správné fungování celého úseku. První pozice testů přidělí jednotce první volný test. Pokud žádný volný není, jednotka čeká, dokud se nějaký neuvolní. Poslední pozice testů se stará o ukončení testování a o dodržení FIFO pořadí dotestovaných jednotek.

V části *Kimitsu*2 se nachází dva samostatné testovací úseky (prvním jsou pozice 95, 97, 99, 101, 103, 105 a 107, druhým jsou pozice 111, 113, 115, 117). Jednotka při průjezdu částí využije testovací stanice v obou úsecích. Protože ale z databáze je možné získat pouze čas vstupu a opuštění části jako celku, nejsou dostupné údaje o tom,

v kterém testovacím úseku se jednotka aktuálně nachází. Proto se oba úseky spojí do jednoho a pro zachování číslování pozic se z pozic s čísly 108 a 109 stane testovací dvojpozice.

Zároveň kvůli rozšíření testovacího úseku o první a poslední testovací pozici se musí posunout křižovatka číslo 92, a s tím i začátek části *Kimitsu*2. Jelikož se jedná o nepatrnou měnu a její vliv je zanedbatelný.

Poslední částí linky je *Mol*, kde probíhá třízení jednotek. Tato část přesahuje model projektu, a proto se jednotky, které se v ní nacházejí, mohou označit jako dokončené. Pozice patřící do části *Mole*, tj. pozice 211 a dál, se vypustí z modelu.

Víše zmíněné změny jsou znázorněny na obrázku 3.1.

Obrázek 3.1. Analýza linky R5 rozdělené na dva úseky s vyznačenými novými strukturami po úpravě modelu

3.2 Vybrané technologie

Projekt bude využívat technologií ASP.NET MVC [8]. Je to bohatá architektura pro vytváření webových aplikací a rozhraní API pomocí návrhového vzoru Model-View-Controller. O grafické zobrazení se postará 3D knihovna Three.js [9], pomocí které se snadno vytváří 3D obsah na webové stránce. Dále se použije knihovna SignalR [10], což je nástroj od Microsoftu, který umožňuje asynchronně posílat zprávy mezi serverem a klientem. Následně bude využita verze lokální databáze SQL Server Express LocalDB [11] pro práci s daty.

3.3 Návrh uživatelského rozhraní

Tato kapitola se zabývá návrhem uživatelského rozhraní podle metody UCD. Metoda spočívá v přesné specifikaci potřeb uživatele, na základě kterých byl vypracován návrh řešení. Návrh byl následně diskutován se zadavatelem práce a příslušně upraven. Výsledný návrh řešení je blíže rozebrán v kapitole ?? Návrh řešení.

3.3.1 User-Centered Design

Podle knihy *User-Centered Design* [12] je UCD metoda používaná vývojáři a designery, aby zajistili vytvoření produktu, který splní požadavky uživatele. Narozdíl od HCI, které se soustředí na interakci člověka s aplikací, se zaměřuje čistě na potřeby uživatele. Použití UCD metody šetří čas tím, že pomáhá předcházet chybám.

3.3.2 Uživatelské požadavky

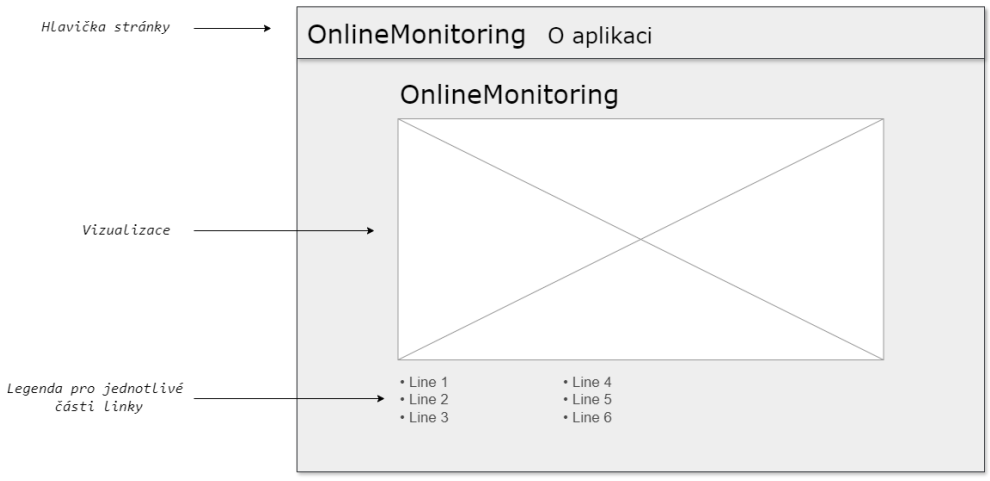
Níže jsou vybrány úkony cílových uživatelů, tedy procesních inženýrů, kteří pomocí aplikace budou v DICZ řešit problémy s výrobou.

- Dohledání části linky, kde se jednotka nachází.
- Snadné rozpoznání jednotlivých částí linky.
- Dohledání potřebných informací k jednotce.
- Zobrazení jednotek v opravě.

3.3.3 Návrh řešení

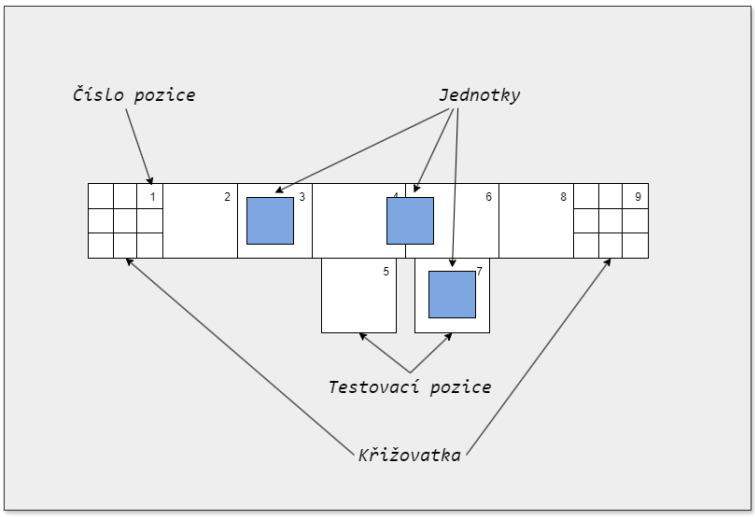
Webová stránka bude obsahovat hlavičku s dvěma odkazy.

První stránka bude zobrazovat výsledek projektu, kde hlavním prvkem aplikace bude grafická vizualizace, proto by měla zabírat většinu stránky. Pod ní bude legenda pro jednotlivé části linky. Návrh webové stránky znázorňuje obrázek 3.2.



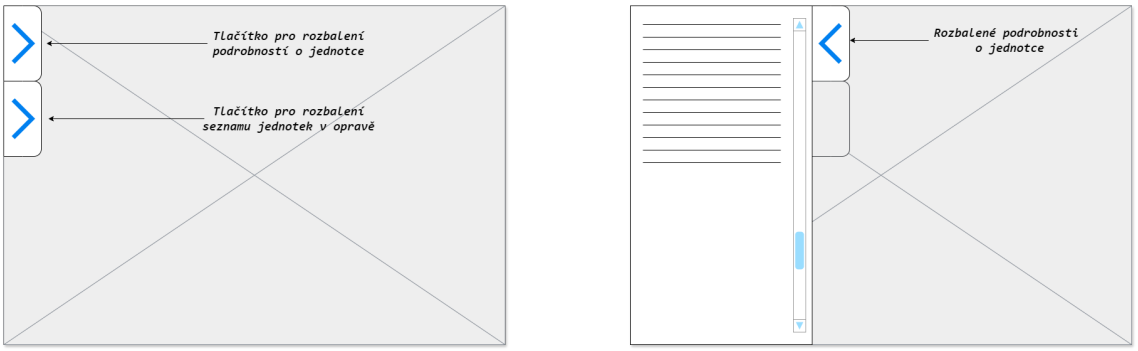
Obrázek 3.2. Návrh rozložení webové stránky

Samotná vizualizace bude zobrazovat pozice označené čísly. Jediná z pozic, kterou je potřeba graficky rozlišit kvůli odlišnému chování od ostatních, je křižovatka. Ta bude zvýrazněna mřížkou. Další z pozic s odlišných chováním je testovací pozice, kterou není potřeba nijak zvýrazňovat, je snadno rozeznatelná při prvním pohledu na rozložení pozic, protože vybočuje z hlavního směru linky (znázorněno na obrázku 3.3).



Obrázek 3.3. Návrh zobrazení vizualizace

Po pozicích se budou pohybovat jednotky, u kterých se při přejetí kurzoru myši zobrazí jejich identifikační údaje. Protože bude potřebovat zobrazovat více podrobné informace o jednotce (až 100 různých dat), bude použita schovatelná záložka. Bude tedy na uživateli, jestli si nechá informace zobrazit, nebo je schová, aby mu neblokovaly vizualizaci. Jednotka, které se podrobnější informace budou týkat, bude zvolena kliknutím levého tlačítka. Návrh je vidět na obrázku 3.4.



Obrázek 3.4. Návrh záložky pro podrobnější informace, zobrazené v okně pro vizualizaci

Jednotky, které se nacházejí v opravě, budou vypsány v další schovatelné záložce. Protože takové jednotky se nenacházejí na lince, nejsou vyobrazeny ve vizualizaci, ale stále je nutné informovat o tom, kde se nacházejí.

Druhý odkaz v hlavičce stránky bude obsahovat popis a ovládání aplikace.

Kapitola 4
Popis řešení

V této kapitole budou představeny dvě aplikace, které vznikly v rámci této práce. Podrobně se rozebere jejich implementace a poté následují ukázky aplikací.

4.1 Aplikace na ovládání databáze

Pro snadné ovládání databáze v rámci testování byla vytvořena WPF aplikace na její správu s názvem ManageDB. Umožňuje manuálně přidat novou jednotku, poslat jednotku do další části, označit jednotku jako dokončenou, příp. jako NG, nebo vrátit jednotku z opravy na vybranou část linky. Také poskytuje automatický režim, kdy se zmíněné úkony vykonávají v pravidelné smyčce.

4.1.1 Implementace

Je dostupných 310 reálných záznamů jednotek, které jsou uloženy v tabulce *IdData*. V případě, kdy je databáze bez dat, aplikace ManageDB je do ní při spuštění nahraje z uloženého Json souboru (*ManageDB/AllUnits.json*).

Data nové jednotky do tabulky *PartKaishi* se vytvoří podle záznamu z *IdDat*, kde se vybere taková jednotka, která následuje po poslední přidané jednotce na lince (pokud taková existuje). Pokud poslední přidaná jednotka na lince je poslední záznam z *IdDat*, vybere se záznam první jednotky. Na lince v jednu chvíli může být maximálně 207 jednotek (linka má 237 pozic, 30 z toho je TestCross, na kterých se jednotka nezdržuje) a některé další jednotky (řádově pár) můžou být na opravě. 310 jednotek je bezpečný počet záznaů, aby se záznamy z jednotek z *IdDat* mohli cyklit a zároveň se neopakovali.

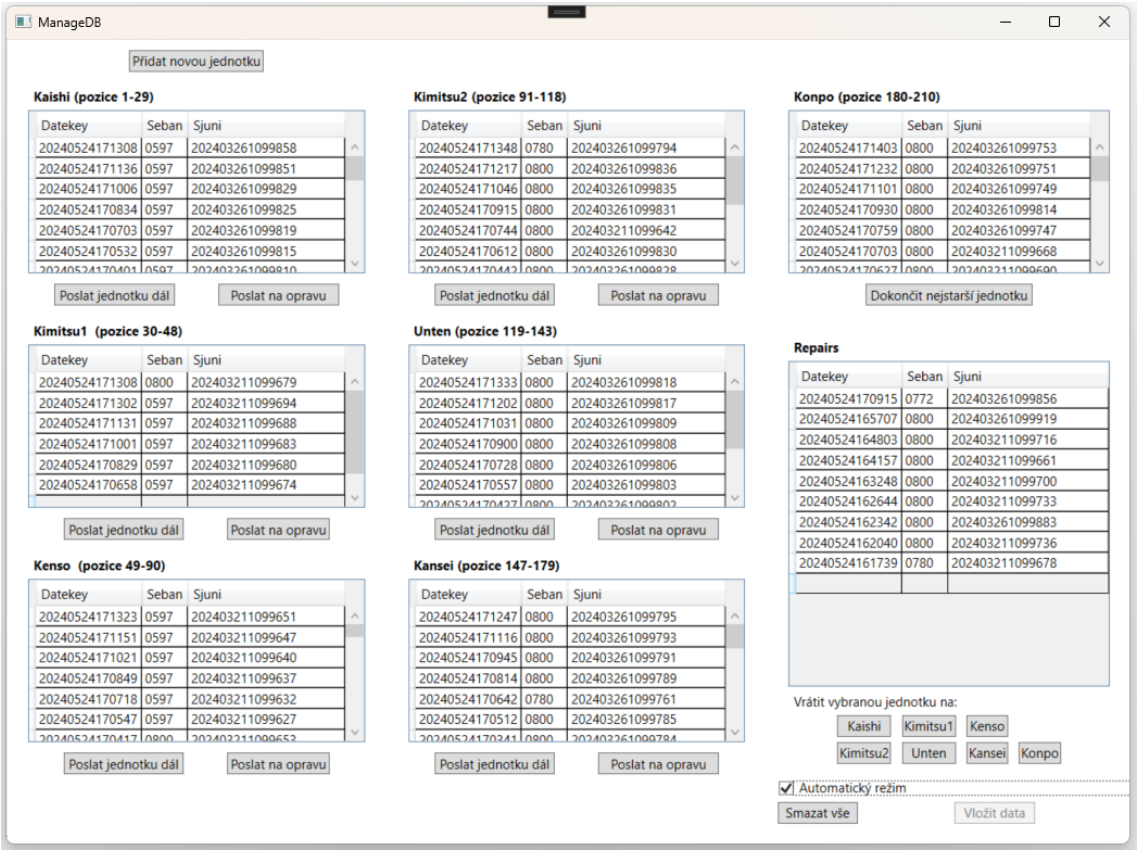
Pokud je linka prázdná, je možné do ní hromadně přidat jednotky pomocí tlačítka 'Vložit data'. Na každou část se přidá počet jednotek podle hodnoty v souboru *ManageDB/MainWindow.xaml.cs*, kde jsou definované ideální počty jednotek pro jednotlivé části. V automatickém režimu se aplikace snaží dodržovat tyto hodnoty.

Pokud je jednotka označena jako NG, aplikace v tabulce *IdData* změní její atribut *NG_Code* na náhodné číslo mezi '1000'-'7999', aby simulovala kód chyby, kvůli které jednotka opustila linku. Kódy chyb jsou rozděleny podle částí linky, kde čísla v rozmezí jednoho tisíce jsou pro jednu část. Stejné číslo je použito i v záznamu jednotky v tabulce *Repairs* v atributu *NG_Code*. V moment, kdy se jednotka vrací z opravy zpět na linku, je tato hodnota v *IdDatech* vrácena na původní kód '0000', tzn. bez chyby.

4.1.2 Ukázka aplikace

V aplikaci (obrázek 4.1) je tabulka pro každou část linky, kde je možné nejstarší jednotku v části buď poslat do další části, nebo na opravu (případně dokončit jednotku, nebo přidat novou). Pro tabulku *Repair* je nutné nejdříve vybrat jednotku, s kterou se bude manipulovat, a následně zvolit tlačítkem, do jaké části se má jednotka vrátit. Pokud je zvolená možnost 'automatický režim', aplikace s jednotkami pohybuje na základě definovaných konstant. Tlačítko 'Smazat vše' smaže všechny záznamy jednotek

ze všech částí linek, i z *Repairs*. 'Vložit data' umožňuje hromadně vložit jednotky do částí.



Obrázek 4.1. Aplikace na ovládání databáze. V tabulkách jsou zobrazeny jednotky nacházející se v jednotlivých částech, je možné s nimi manipulovat pomocí tlačítek.

4.2 Webová aplikace vizualizace

Webová aplikace využívající architekturu ASP.NET MVC, která v reálném čase zobrazuje vizualizaci výrobní linky a zpracovává dotazy od klienta, např. vyhledání podrobností o jednotce.

4.2.1 Implementace

Aby model aplikace byl shodný pro všechny klienty, je koncipován jako singleton. Při jeho vytváření se načte definovaný layout a spustí časovač s definovanou frekvencí (obě hodnoty jsou definovány jako konstanty v *Models/Constants.cs*).

V rámci časovače se zavolá tik modelu, který se skládá celkem ze tří částí. První se stará o aktualizaci dat z databáze, další o vyhodnocení pohybu jednotek na pozicích a na závěr se nová data pošlou všem klientům v Json formátu.

Tvoření layoutu je podrobně zpracováno v příloze B.2.1 - Tvoření Layoutu. Layout se tvoří od poslední pozice k první a u každé pozice se definuje její třída, rozměr, číslo, relativní pozice vzhledem k vedlejší pozici a část, do které patří. Layout je po spuštění aplikace vycentrován a uživateli je omezen pohyb kamery mimo layout a zároveň se jednotky rovnoměrně rozloží na lince.

Data z databáze se neaktualizují každý tik (aktualizují se každý x-tý, hodnota je opět definována v konstantách). Protože jednotce trvá zhruba 50s, než přejede pozici,

není potřeba tak častá aktualizace dat, jako u vyhodnocení pohybu jednotek. Nejdříve se stáhnou data z tabulek všech částí a následně se jednotlivé jednotky porovnávají s jednotkami, které jsou již uloženy v modelu. V případě nějaké změny u jednotky (hlavně změna části a *DateKey*) se data aktualizují a informuje se nejbližší následující křižovatka o příjezdu jednotky. Tato informace pomáhá křižovatce dodržovat FIFO pořadí jednotek. Pokud se jedná o úplně novou jednotku, je přidaná na začátek linky. Pokud je jednotka již označena jako hotová a zároveň už zmizela z linky, je její záznam odstraněn z modelu.

Vyhodnocování pohybu se narozdíl od aktualizace dat provádí každý tik, aby byl pohyb jednotek co nejplynulejší. Hlavní logika je celá řešena na straně serveru, a proto aby se zabránilo vysoké frekvenci posílání zpráv ke klientům, je na straně klienta provedena interpolace mezi jednotlivými zprávami. Díky tomu plynulost pohybu jednotek nebude závislá na frekvenci posílání zpráv.

Jednotka sama o sobě nemá žádnou logiku, je to objekt, který si předávají pozice. Její definice je ve výpisu 4.1.

```
public class Unit
{
    public string UnitName { get; }
    public string DateKey { get; set; }
    public string SequenceNumber { get; }
    public string NG_Code { get; set; }
    public MyVector2 Center { get; set; }
    public PartOfLine PartOfLine { get; set; }
    public float TransitionParameter { get; set; }
    public UnitState Tag { get; set; }
    public int ActualTestPosNumber { get; set; }
    public float Opacity { get; set; }

    public Unit(string seban, string dateKey, string sjuni,
                PartOfLine partOfLine, string ng_code = "0000")
    {
        UnitName = seban;
        DateKey = dateKey;
        SequenceNumber = sjuni;
        Center = new MyVector2(0f, 0f);
        PartOfLine = partOfLine;
        Opacity = PartOfLine == PartOfLine.Finished ||
                  PartOfLine == PartOfLine.NG ?
                    0f : 1f;
        NG_Code = ng_code;
    }
}
```

Výpis 4.1. Třída *Unit*

Proměnné jednotky *UnitName* (neboli *Seban*), *SequenceNumber* (neboli *Sjuni*) jsou identifikátory jednotky. *PartOfLine* určuje část linky, v které se jednotka aktuálně nachází a *DateKey* je čas vložení jednotky do tabulky části a pomáhá určovat pořadí jednotek na lince. *NG_Code* má defaultní hodnotu '0000', která značí jednotku bez chyby, pokud se jednotka nachází na lince. Pokud je jednotka v opravě, její hodnota *NG_Code* bude jiná a bude získaná z tabulky *Repairs*.

Proměnné *Center* a *Opacity* slouží k zobrazování jednotky. *Center* určuje střed jednotky a je typu *MyVector*2. Tato třída kopíruje základní vlastnosti třídy *Numerics.Vector* jako x a y pozici, sčítání a násobení vektorů a rozšiřuje ji o snadnou serializaci. Třída *Numerics.Vector* nemůže být serializována pomocí Json, v jehož formátu se posílají data mezi serverem a klientem. *Opacity* určuje průhlednost jednotky. Ta se mění případech, kdy se jednotka objevuje na lince nebo mizí z linky.

Pozice jednotky, tedy hodnota *Center*, je vypočítána pomocí proměnné *TransitionParameter*. Třída pozice definuje dráhu jednotky a *TransitionParameter* je hodnota mezi 0 a 1, která určuje pozici jednotky mezi koncovými body dráhy.

Zbývající proměnné jsou potřebné pro testovací úseky. *ActualTestPosNum* je aktualizována první testovací pozicí (FirtTestPosition) a určuje přiřazenou testovací pozici jednotce. *Tag* určuje stav jednotky a její hodnoty jsou následující:

- Basic - defaultní hodnota
- Testing - jednotky, které zajíždí do přiřazené testovací pozice
- Crossing - jednotky, které již jsou dotestované a projíždí testovacím úsekem do další části

Každá pozice má vlastní číslo, které není nutně jedinečné. Pozice rozhodují o pohybu jednotek. Proměnné třídy Position jsou vypsány ve výpisu 4.2.

```
public int PositionNumber { get; }
public MyVector2 Measures { get; }
public MyVector2 Center { get; set; }
public PartOfLine PartOfLine { get; }
public PositionType Type { get; }
public virtual Position? NextPosition { get; protected set; }
public List⟨Transition⟩ Transitions { get;  set; }
public Unit? ControlUnit { get; set; }
protected static float MaxEps = 0.01f;
```

Výpis 4.2. Proměnné třídy *Position*

PositionNumber je zmíněné číslo pozice. *Center* a *Measures* určují umístění a rozměr pozice. *PartOfLine* je hodnota enumu definující část linky, do které pozice patří a *Type* je enum definující typ pozice, tj. křižovatka, první testovací pozice, atd... *NextPosition* je odkaz na následující pozice, které se bude předávat jednotka nacházející se na aktuální pozici, která je uložena v *ControlUnit*. *Transitions* je seznam drah třídy *Transition*, které definují pohyb jednotky po dané pozici.

```
public class Transition
{
    public MyVector2 FromPosition { get; set; }
    public MyVector2 ToPosition { get; set; }
    public float FromTransitionParameter { get; }
    public float ToTransitionParameter { get; }
    public UnitState Tag { get; }
}
```

Výpis 4.3. Proměnné třídy *Transition*

V rámci *Transition* (výpis 4.3) se pozice bere jako jednotkový čtverec s bodem [0, 0] ve středu. *FromPosition* a *ToPosition* jsou vektory, které určují kudy dráha vede. *FromTransitionParameter* a *ToTransitionParameter* určují, pro jakou hodnotu *TransitionParameter* jednotky je dráha určená a *Tag* určuje hodnotu *Tagu*

u jednotky. *Tag* je důležitý zejména u testovacích křižovatek, kdy podle *Tagu* jednotka zajede buď do testovací stanice, nebo pojede dál po lince.

Pro rohovou pozici, kde jednotka pojede nejdříve zleva doprava a poté, co dosáhne středu pozice, pojede nahoru, je definice *Transitions* znázorněna ve výpisu 4.4.

```
Transitions = new() {
    new Transition(
                fromPosition: Direction.Left,
                toPosition: Direction.Right,
                fromTransitionParameter: 0f,
                toTransitionParameter: 0.5f,
                tag: UnitState.Basic);
    new Transition(
                fromPosition: Direction.Bottom,
                toPosition: Direction.To,
                fromTransitionParameter: 0.5f,
                toTransitionParameter: 1f,
                tag: UnitState.Basic);
}
```

Výpis 4.4. Definice *Transition* pro rohovou pozici

Enum *Direction* je v konstruktoru převeden na vektor $MyVector2$.

Proměnná *MaxEps* zabraňuje jednotkám přejíždět pozice tím, že omezuje maximální rozdíl mezi *TransitionParameter* jednotky a *TransitionParameter* určeným dráhou.

V konstruktoru pozice je proměnná *relativeLocationToNextPosition* třídy *Direction*, pomocí které je vypočítána přesná pozice jednotky v závislosti na relativním umístění k následující pozici a rozměrech obou pozic. Zároveň se automaticky vypočítá *Transitions* pozice, případně se pozmění *Transition* následující pozice, pokud je pozice rohová.

Na každé pozici se může nacházet pouze jedna jednotka. Pokud se jednotka nemůže z nějakého důvodu pohybovat dál, vyčkává ve středu pozice. Speciální pozice rozšiřují funkcionalitu obyčejné pozice.

- Position: Posune jednotku o definovanou rychlost v závislosti na čase od posledního tiku po dráze, pro kterou jednotka splňuje podmínky. Pokud jednotka patří do následující části, než je aktuální pozice, jednotka se pohybuje zvýšenou rychlostí, aby změnu co nejdříve vyrovnala. Pokud následující pozice patří do jiné části než aktuální pozice, jednotka vyčkává, dokud se její záznam neobjeví v následující části. Pokud se na následující pozici nachází jiná jednotka, aktuální jednotka není poslána dál a čeká, až následující pozice bude volná. V případě, že střed jednotky přesahuje pozici, kontrolu nad jednotkou přebírá následující pozice.
- CrosswayPosition: Na pozici se můžou jednotky objevovat nebo mizet (jednotky, které jdou z opravy na linku nebo z linky na opravu, případně nové nebo dokončené jednotky). Mizející a objevující se jednotky jsou rozlišeny barevně. Pokud je pozice poslední pozicí na lince, jednotku neposílá na další pozici (která neexistuje), ale nechá jednotku zmizet, pokud je jednotka označena jako dokončena (tj. nachází se v tabulce *PartMol*). Pozice si hlídá pořadí jednotek podle *DateKey* a podle toho vyhodnocuje jejich pohyb/objevování/mizení.
- FirstTestPosition: Pozice si udržuje seznam testovacích stanic, které k ní náleží. Jednotce přiřadí první volnou stanici a její číslo uloží do proměnné jednotky *ActualTestPosNum*. Pokud žádná stanice volná není, jednotka vyčkává.

- TestCross: Pozice má dva stavy: volno a testuje. Testuje v případě, kdy se na příslušné testovací pozici nachází jednotka, jinak má stav volno a může jí být přiřazena jednotka. Pokud na pozici dorazí jednotka, která je přiřazena k příslušnému testu, je *Tag* jednotky změněn na *Testing* a jednotka je odkloněna do testovací stanice. Jednotky s ostatními hodnotami *Tagu* pošle dál.
- TestPosition: Jednotka na pozici čeká, dokud její testování není dokončeno, tj. její *Tag* byl změněn na *Crossing*. Pokud se tak stane, jednotka si zkontroluje pozice ostatních jednotek na dané části a vyjede z pozice, případně čeká, aby opustila testovací část ve správném pořadí.
- LastTestPosition: Pozice se stará o dokončení testů tím, že pro dotestované jednotky (jednotky, jejichž záznam je již v dalšá části) mění jejich *Tag* na *Crossing*.

Zpráva v tiku se pošle v Json formátu všem klientům a zpráva se skládá ze dvou seznamů. První jsou aktualizovaná data jednotek po vyhodnocení jejich pohybu na pozicích, druhý obsahuje jednotky, kterým se mění průhlednost, tj. nové jednotky, dokončené jednotky, nebo jednotky jdoucí na opravu nebo z opravy. Takové jednotky jsou při zobrazování, kvůli speciálním změnám, zpracovány samostatně. Jednotky jdoucí na opravu se přidají do schovatelné záložky, kde je seznam všech jednotek v opravě, jednotky jdoucí z opravy na linku se ze seznamu odeberou.

Na straně klienta se následně zpráva zpracuje. Data se uloží do hashovací tabulky, kde je pro každou jednotku zpráva uložena dvakrát. Jedna aktuální a druhá předešlá, kdy ta předešlá se vždy přepíše nově příchozí a z aktuální se stane předešlá. Následně se mezi těmito zprávami interpoluje umístění a průhlednost jednotky a zajistí se tím plynulý pohyb jednotek. Interpolace je závislá na čase tiku na straně klienta a na čase mezi zprávami.

Obrázek jednotky je načten z SVG souboru a zpracován do křivek knihovny three.js. Pro různé *Sebany* patří různé obrázky a jsou určeny pomocí slovníku *SVGDict* ve třídě *SVGSetUp*, kde klíč je název SVG souboru a hodnota je seznam příslušných *Sebanů*. Na lince R5 se vyrábí několik modelů jednotek, které se dají rozdělit na 3 hlavní druhy, které využívají 3 různé obrázky.

Po přejetí kurzorem po jednotce se zobrazí její identifikáční údaje, tj. *Seban* a *Sjuni*. Poté, co kurzor jednotku opustí, informace o jednotce zase zmizí. V případě, že uživatel chce zobrazit podrobnosti o jednotce, je potřeba na ní kliknout. V ten moment se pošle dotaz na server, který vybere příslušná data z tabulky *IdData* podle kombinace *Sebanu* a *Sjuni*. Poté k nim připojí, pokud existují, data z tabulky *Hinshitsu* (tabulka pro jednotky, které jedou linkou podruhé), tentokrát ale podle kombinace *Sebanu* a *Kibanu* (*Kiban* je získaný z tabulky *IdData*). Každá z tabulek má přes 100 atributů a ne všechny je potřeba zobrazovat, proto jsou ve třídě *UnitDBInfoStruct* definovány seznamy atributů, které se mají z tabulek zpracovat. Seznamy používají třídu *InfoStruct*, jejíž definice je ve výpisu 4.5.

```
public class InfoStruct
{
    public string ColumnName { get; set; }
    public string? Value { get; set; }
    public string Description { get; set; }
    public bool IsRST { get; set; }

    public InfoStruct(string colName, string des, bool isRst = false)
    {
```

```
            ColumnName = colName;
            Description = des;
            IsRST = isRst;
            Value = "";
        }
    }
```

Výpis 4.5. Třídy *InfoStruct*

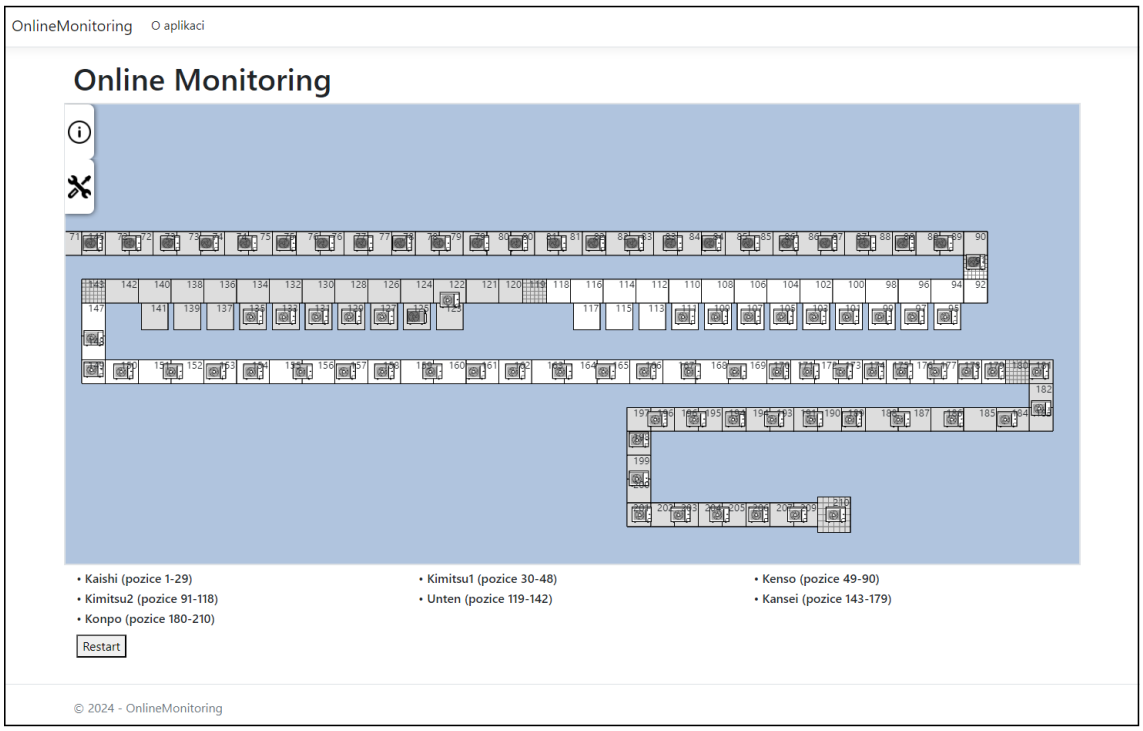
ColumnName a *Value* jsou jméno atributu a jeho hodnota z tabulky, *Description* je popis, pod kterým je hodnota následně zobrazena a *IsRST* je příznak, který pokud je pravdivý, tak hodnotě přiřadí speciální formát:

- ’00’ - šedé pozadí - defaultní hodnota
- ’01’-’09’ - červené pozadí - chybový kód
- ’10’-’19’ - zelené pozadí - výsledek OK
- jinak pozadí oranžové - pouze testovací data

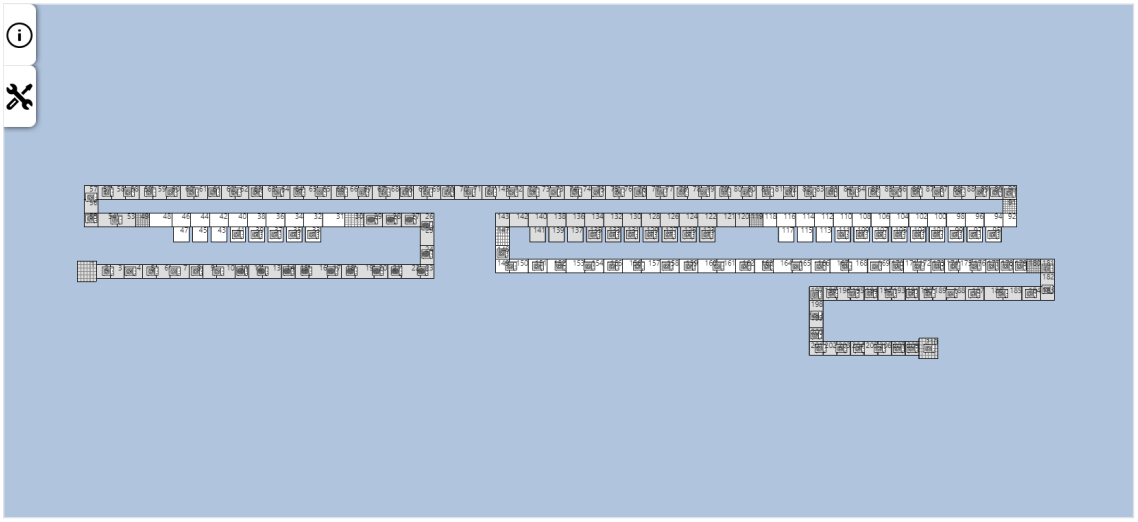
Data se následně odešlou zpět klientovi a zobrazí se ve schovatelné záložce. Pro jednotky v opravě je proces stejný, jen je potřeba kliknout na tlačítko pro informace pro příslušnou jednotku. Seznam jednotek v opravě se nachází ve vedlejší schovatelné záložce.

4.2.2 Ukázka aplikace

Většinu webové aplikace (obrázek 4.2) vyplňuje zobrazení vizualizace (obrázek 4.3). Pod ní se nachází legenda s rozložením částí. Jednotlivé části linky jsou graficky rozlišeny. Každá pozice je označena svým identifikačním číslem.

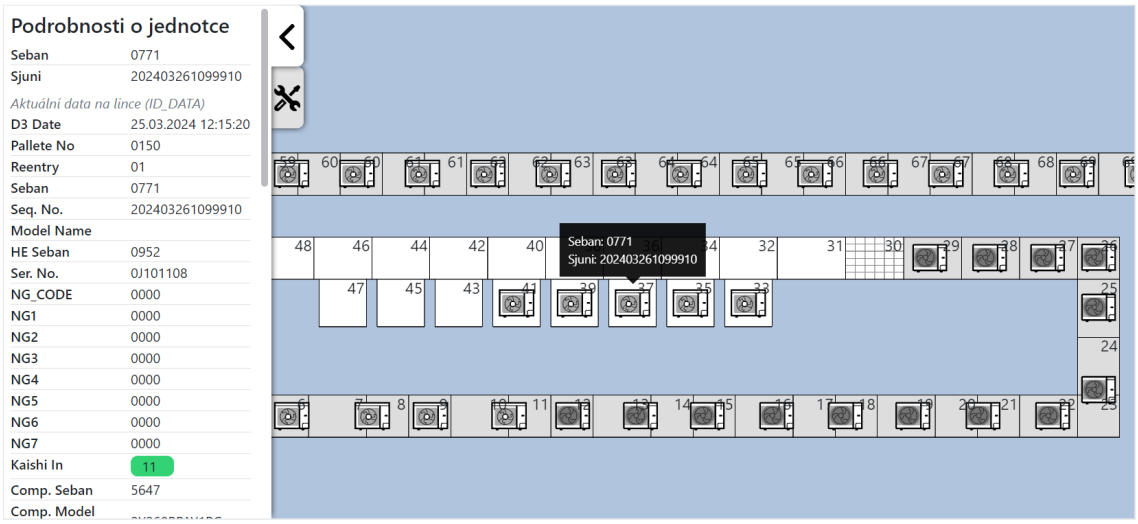


Obrázek 4.2. Webová aplikaci vizualizace. Hlavní část zabírá vizualizace, pod ní se nachází legenda s rozložením částí. Tlačítko ’Restart’ zajišťuje okamžitý propis změn v databázi.



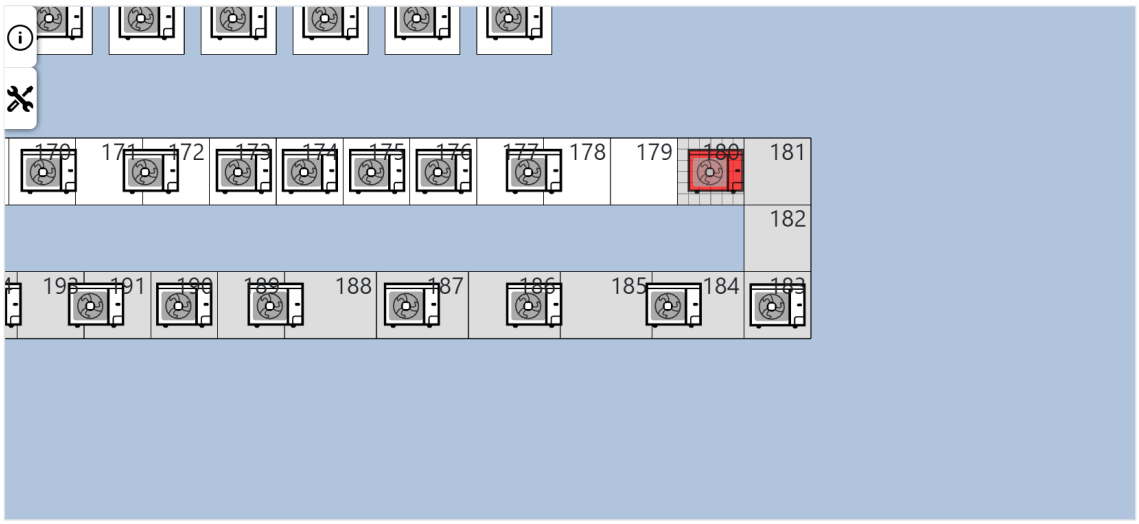
Obrázek 4.3. Detail na vizualizaci s pohledem na celou linku. Jednotlivé části linky jsou graficky rozlišeny. Každá pozice je označena svým identifikačním číslem.

Pro zobrazení identifikačních údajů o jednotce (*Sebanu* a *Sjuni*) je potřeba na jednotku najet myší. Pro bližší infomace je potřeba na jednotku kliknout levým tlačítkem, kdy se informace zobrazí v levé postranní schovatelné záložce (obrázek 4.4).

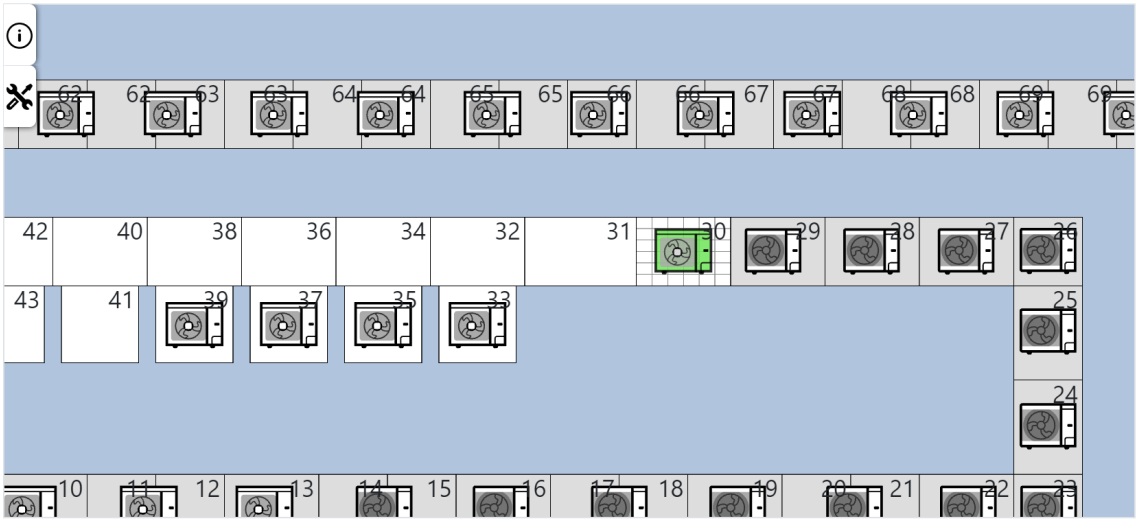


Obrázek 4.4. V tmavém obdélníku jsou zobrazené identifikační údaje o jednotce pod ním, na levé straně jsou zobrazeny podrobnosti o jednotce. Hodnota $KaishiIn$ využívá speciální formátování $IsRST$.

Každá křižovatka je graficky označena šedou mřížkou. Jednotky na ní se objevující/mizející jsou barevně rozlišeny podle stavu. Červená jednotka z linky mizí (obrázek 4.5), její stav je NG. Zelená jednotka dokončila opravu a vrací se na linku (obrázek 4.6).

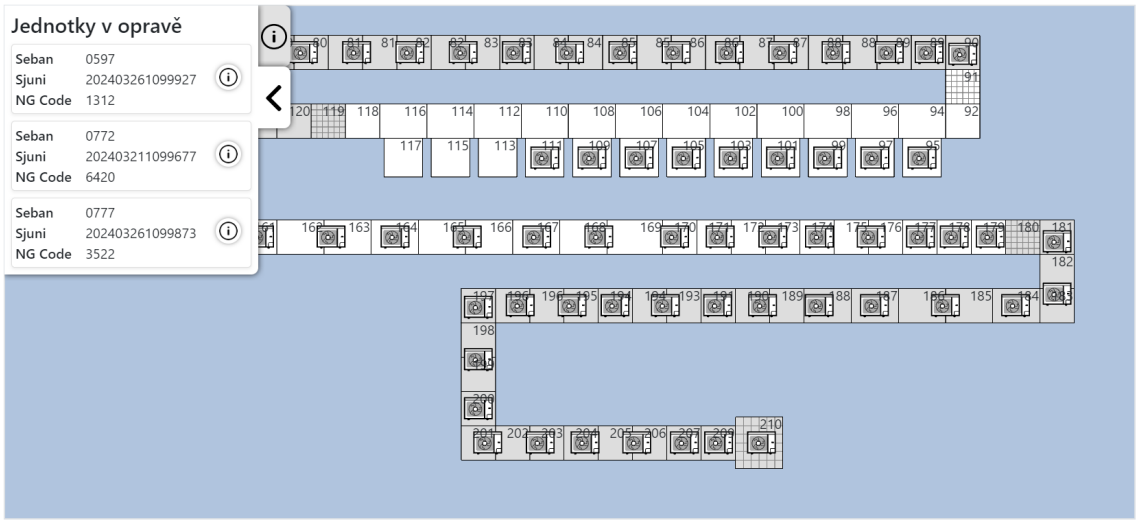


Obrázek 4.5. Jednotky odjíždějící z linky na opravu na křižovatce (pozice s šedou mřížkou) zmizí. Při mizení jsou obarveny na červeno.



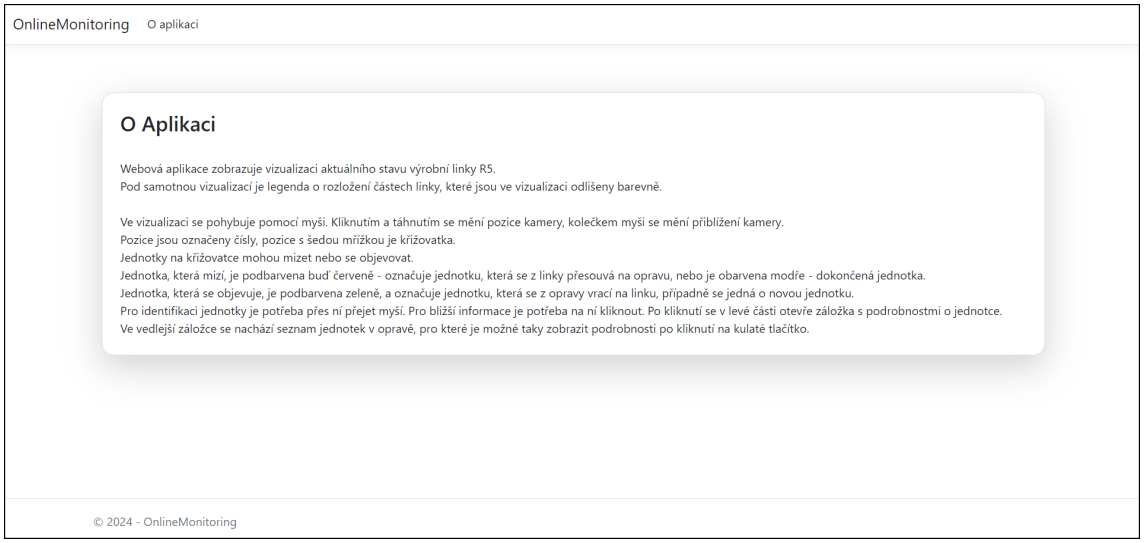
Obrázek 4.6. Jednotky přijíždějící na linku z opravy se na křižovatce (pozice s šedou mřížkou) objeví. Při objevování jsou obarveny na zeleno.

Jednotky, které jsou na opravě, se zobrazují v postranní schovatelné záložce (obrázek 4.7). Pro zobrazení podrobností o jednotce je potřeba kliknout na kulaté tlačítko 'i'.



Obrázek 4.7. Seznam jednotek v opravě v záložce nalevo. Pro podrobnosti o jednotce je potřeba kliknout na tlačítko 'i'.

Nad druhé stránce webové aplikace se nachází popis aplikace a návod na její ovládání (obrázek 4.8).



Obrázek 4.8. Záložka s popisem webové aplikace

Kapitola 5
Uživatelské testování

V této kapitole je popsáno, jak byla webová aplikace vizualizace testována, jakým způsobem testování probíhalo a jaké jsou výsledky testování.

Cílovou skupinou jsou zaměstnanci DICZ, zejména procesní inženýři, kteří se starají o správnou funkčnost linky.

Testování aplikace proběhlo s pěti účastníky. Všichni jsou zaměstnanci DICZ a jsou obeznámeni s problematikou. První z nich je IT technik, který nebude aplikaci využívat. Další dva jsou mistři linky, kteří aplikaci budou používat na sledování linky, pokud se u ní nebudou nacházet, a poslední dva jsou procesní inženýři, kteří aplikaci budou využívat pro řešení problémů na lince.

5.1 Průběh testování

Test probíhal v izolované místnosti, kde se nacházel jeden z účastníků a já, jako moderátor. Účastníkům byla představena aplikace, její cíle a účel testování. Poté měli čas si aplikaci prohlédnout a vyzkoušet a až byli připraveni, byli dotazováni na testovací otázky, jejichž plná podoba je dostupná v příloze C - Otázky z uživatelského testování. Odpovědi jsem si pečlivě zaznamenávala.

Otázky testovaly orientaci v aplikaci, pochopení provedených změn oproti reálnému rozložení linky a nakonec zjišťovali, co účastníkům v aplikací chybí a co by chtěli změnit.

Jeden test trval zhruba 10-20 minut.

5.2 Výsledky testování

Odpovědi na jednotlivé otázky jsou následující:

- Ukaž v aplikaci na úsek linky *Kansei*?
  - Všichni účastníci dokázali správně ukázat na pozice 143 - 179.
- Je grafické rozdělení částí jednoznačné/dostatečné?
  - Jeden z mistrů by preferoval odlišnou barvu pro každou část, šedo-bílé rozdělení mu přijde nevýrazné.
  - Druhý z mistrů ocenil legendu pod vizualizací.
  - Ostatní účastníci byli s obarvením spokojeni.
- Najdi sériové číslo jednotky, která se testuje na pozici 105.
  - Všichni účastníci dokázali najít hledanou jednotku a zobrazit si její podrobnosti, v kterých našli sériové číslo.
  - IT technik by ocenil možnost filtrovat data mezi podrobnostmi.
- Kolik jednotek se nachází v opravě?
  - Všichni účastníci dokázali správně rozkliknout záložku s jednotkami v opravě.
  - IT technik by preferoval zobrazení celkového počtu jednotek.

- Proč jsou pozice 30, 49, 91, 119, 143, 180 odlišné?
  - IT technik nerozpoznal, že pozice jsou křižovatky, ale všimnul si, že se jedná o počáteční pozice částí.
  - Jeden z inženýrů blízce specifikoval, že se jedná o křižovatku, neboli datový konec a začátek částí linky. Tedy místo, kde se záznam jednotky přesouvá z jedné tabulky části do další.
  - Ostatní odpovědeli jednoduše, že se jedná o křižovatky.
- Co znamená mizení jednotek na pozici 49?
  - IT technik po vysvětlení, že pozice je křižovatka, odpověděl, že jednotka jede na kontrolu kvality.
  - Ostatní odpověděli, že se jedná o NG jednotku, která nejspíš neprošla testy, a musí jet na opravu.
  - Obě odpovědi jsou správné.
- Jsou obrázky jednotek dostatečné?
  - Jeden z mistrů poznamenal, že použití jiných obrázků podle modelů není potřeba. Stačil by jeden unifikovaný.
  - Jeden z procesních inženýrů by chtěl jiný obrázek pro každý model.
  - Ostatní byli spokojeni.
- Je něco, co by si na aplikaci změnil/přidal/odebral?
  - Jeden mistr a inženýr by preferovali pozice nahradit reálným layoutem, druhému inženýrovy by stačilo použít layout pouze jako pozadí, aby to usnadnilo orientaci.
  - Jednomu z inženýrů přijdou křižovatky nejednoznačné a přidal by k nim vedlejší pozice.
  - Druhý z mistrů by uvítal viditelný počet jednotek na celé lince a jednotlivých částí.

Pro přidání vedlejších pozic ke křižovatkám by bylo potřeba vytvořit novou třídu pozice, která by neměla žádnou logiku, a zároveň by to samozřejmě zesložitělo tvoření modelu layoutu. Tomuto řešení bych se ráda vyhla, protože se domnívám, že by uživatel očekával, že se jednotka bude po vedlejších pozicích pohybovat, stejně jako se děje na skutečné lince. Řešením může být zobrazení dvou šipek vedle křižovatky, jedné červené směřující ven a druhé zelené směřující dovnitř, které by uživateli napovídaly i význam barev mizejících/objevujících se jednotek.

Poznámka o nahrazení pozic reálným layoutem by byla možná. Stačilo by zrušit vykreslování pozic a použít obrázek layoutu jako pozadí, ale přesto by vytvoření modelu layoutu bylo stále potřeba, protože definuje dráhu jednotky.

Zmíněné další návrhy na úpravy (filtr, zobrazení počtu jednotek, podbarvení pozic) jsou drobnosti, které se dají snadno upravit na míru při nasazování aplikace do systému DICZ.

Kapitola 6
Diskuze

Cílem této práce byl návrh a implementace webové aplikace zobrazující aktuální stav výrobní linky. Aplikace má pomoci zaměstancům při řešení problémů, jelikož systém má zpřehledňovat a vizualizovat data uložená v mnoha složitých tabulkách v databázi. Po analýze problému jsem vybrala vhodné technologie a navrhla řešení. Posléze jsem aplikaci naimplementovala a otestovala s vybranými zaměstnanci z DICZ.

V aplikaci je poměrně časově náročná tvorba layoutu linky. Každou pozici je potřeba samostatně definovat, určit její typ, rozměry a část linky, do které patří. Použití aplikace na jinou linku by tedy byla složitá operace, ale alespoň je jednorázová, protože rozložení linek je stálé. V budoucnu by stálo za zvážení proces zjednodušit, ideálně vytvořit editor na jeho vytváření a úpravu.

Jedná se o první verzi aplikace, která vizualizuje pouze jednu linku v DICZ. V dalších verzích je vize aplikaci mimo jiné rozšířit na všech 9 linek. Tato úprava bude vyžadovat vytvoření 9 různých datových modelů aplikace, se kterými budou pracovat samostatná vlákna.

Z uživatelského testování je patrných pár návrhů na změnu. Některé návrhy, které si protiřečí, je možné upravit pro každou linku zvlášť podle požadavků mistrů. Hlavní slovo však mají procesní inženíři, pro které je aplikace primárně určena.

Aplikace je připravena na nasezaní do výrobního systému DICZ a na případné úpravy a rozšíření funkcionalit.

Literatura

[1] *Daikin*. 2024. https://www.daikinczech.cz/cz_cz/index/o-dicz/dicz-plzen.html. [Online] Cit. 2024-05-04.

[2] *Visual Components*. 2024. https://www.visualcomponents.com. [Online] Cit. 2024-05-09.

[3] *Visual Components Premium*. 2024. https://www.visualcomponents.com/products/premium/. [Online] Cit. 2024-05-09.

[4] Jian-Fei Chai, Xiao-Mei Hu, He-Wei Qu, Ming-Hang Li, Hui-Jing Xu, Yu Liu a Tao Yu. Production line 3D visualization monitoring system design based on OpenGL. *Adv. Manuf.* 2018, 6 126-135. DOI https://doi.org/10.1007/s40436-018-0217-x.

[5] Manuel D. Rossetti. *Simulation Modeling and Arena*. 3rd and Open Text Edition vydání. 2021. Dostupé z https://rossetti.github.io/RossettiArenaBook/.

[6] *Arena Simulation Software in Food & Beverage*. 2024. https://www.rockwellautomation.com/content/dam/rockwell-automation/sites/videos/offering/information-solutions/cookie-demo/cookie-demo.png. [Online] Cit. 2024-05-15.

[7] Rafael Maio, Tiago Araújo, Bernardo Marques, André Santos, Pedro Ramalho, Duarte Almeida, Paulo Dias a Beatriz Sousa Santos. Pervasive Augmented Reality to support real-time data monitoring in industrial scenarios: Shop floor visualization evaluation and user study. *Computers Graphics*. 2024, 118 11-22. DOI https://doi.org/10.1016/j.cag.2023.10.025.

[8] *Overview of ASP.NET Core MVC*. 2023. http://typotypo.wz.cz/csn016910.pdf. [Online] Cit. 2023-11-12.

[9] *Fundamentals*. https://threejs.org/manual/#en/fundamentals. [Online] Cit. 2023-11-11.

[10] Viktória Bakonyi, Zoltán Illés, Ildikó Pšenáková a Dávid Szabó. *SignalR based Real-Time applications in education*. In: *2023 3rd International Conference on Innovative Practices in Technology and Management (ICIPTM)*. 2023. 1-6.

[11] *SQL Server Express LocalDB*. 2024. https://learn.microsoft.com/en-us/sql/database-engine/configure-windows/sql-server-express-localdb?view=sql-server-ver16. [Online] Cit. 2024-05-18.

[12] Travis Lowdermilk. *User-Centered Design*. O'Reilly Media, Inc., 2013. ISBN 9781449359836.

Příloha A
Zkratky a slovník

A.1 Zkratky

DICZ	Daikin Industries Czech Republic s.r.o.
R5	Označení výrobní linky v DICZ, pro kterou je aplikace vytvořena
MVC	Model-View-Controller
API	Aplikační programové rozhraní
JSON	JavaScript Object Notation
FIFO	First In, First Out
WPF	Windows Presentation Foundation
UCD	User-Centered Design
HCI	Human-Computer Interaction
NG	Not Good
AR	Augmented Reality

A.2 Slovník

Kishu	Jméno modelu
Sline	Kód linky
Zuban	Výkresové číslo
Seban	Čtyřčíslí nahrazující jméno modelu pro jednodušší kontrolu ve výrobě
Poscode	Šesticiferný kód popisující model jednotky stejně jako Seban
Sjuni	Sekvenční číslo
Kiban	Sériové číslo
Shiji	Výběr (picking)
Skkr - Shikakaru	Výsledek

Příloha B
Uživatelská dokumentace

B.1 Spuštění aplikace

Celá práce je dostupná na GitLabu: `https://gitlab.fel.cvut.cz/kaslonat/monitoring_bp`. Repositář obsahuje následující projekty:

- OnlineMonitoring - webová aplikace vizualizace
- ManageDB - aplikace na ovládání databáze

Pro spuštění je potřeba mít nainstalované Visual Studio 2022.

B.1.1 Inicializace databáze

Pro úspěšné spuštění obou projektů je potřeba lokální OnlineMonitoring databáze. Pokud na počítači není již vytvořena, je potřeba ji inicializovat. Otevřete si projekt OnlineMonitoring ve Visual Studiu. Dále je potřeba otevřít konzoly pro správu balíčků: *Tools* > *NuGet Package Manager* > *Package Manager Console* a spustit následující příkaz:

```
Update-Database
```

B.1.2 Spuštění projektů

Projekty si otevřete ve Visual Studiu a spusťte. V aplikaci ManageDB můžete manuálně přidávat nové jednotky či posílat jednotky dál, nebo můžete zapnout automatický režim. V aplikaci OnlineMonitoring uvidíte vizualizaci jednotek podle záznamů z databáze.

B.1.3 Smazání databáze

Pro smazání lokální databáze z vašeho počítače si znovu otevřete konzoly pro správu balíčků: *Tools* > *NuGet Package Manager* > *Package Manager Console* a spusťte příkaz:

```
Drop-Database
```

B.2 Úprava linky

Níže jsou popsány všechny možné úpravy aplikace, ať už se jedná o rozložení linky nebo zobrazování.

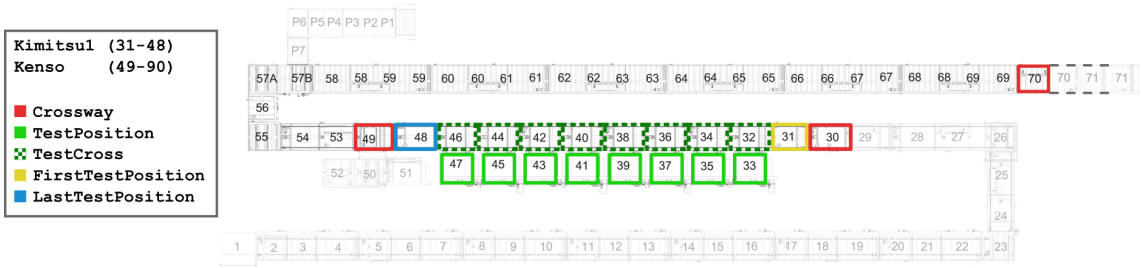
B.2.1 Tvoření layoutu

Jednotky se pohybují po pozicích od nejmenšího čísla k největšímu a tak pozice s menším číslem budou označovány jako první/dřívější.

Jako první je potřeba připravit si rozložení linky, které se bude předělávat do aplikace. Jako příklad bude použit úsek linky R5, přesněji části *Kimitsu*1 a polovina části *Kenso*. Je potřeba si rozmyslet kde začínají a končí jaké části linky a kde budou použity jaké pozice. Pro pozice platí určitá specifika:

- Křižovatka (Crossway)
  - První a poslední pozice celé linky
  - První pozice každé části
  - Na každé části linky (kromě poslední) se nachází pouze jedna křižovatka
- Část testů
  - Testovací křižovatka (TestCross) je taková pozice, která je přilehlá k testu (TestPosition)
  - Před dvojicemi testovacích kžižovatek a testů se musí nacházet první testovací pozice (FirstTestPosition), která patří do stejné části
  - Po dvojicích testovacích kžižovatek a testů se musí nacházet poslední testovací pozice (LastTestPosition), která patří do stejné části

Připravené rozložení předváděného úseku linky je vidět na obrázku B.1.



Obrázek B.1. Připravené rozložení úseku linky R5

Ve Visual Studiu otevřete projekt OnlineMonitoring a najděte složku: *Models/LinesLayout*. Vytvořte novou třídu s názvem vaší linky, např. *R5Layout.cs*. Aby byl váš vytvořený layout aplikací načten, je potřeba přiřadit ho jako aktuálně používaný layout. Návod na změnu konstant naleznete v kapitole B.2.3 - Změna konstant.

Z třídy *NewLayout.cs* zkopírujte závislosti a proměnné. Vaše třída by měla vypadat takto (výpis B.1):

```
using OnlineMonitoring.Models.Positions;
using OnlineMonitoring.Models.Utils;
using static OnlineMonitoring.Models.Utils.Enums;
using static OnlineMonitoring.Models.Utils.DirectionToDict;
using OnlineMonitoring.Models.Positions.TestPosition;

namespace OnlineMonitoring.Models.Lines
{
    public static class R5Layout
    {
        private static readonly MyVector2 SmallPosition =
```

```
                new MyVector2(40f, 40f);
        private static readonly MyVector2 WidePosition =
                new MyVector2(55f, 40f);
        private static readonly MyVector2 SuperWidePosition =
                new MyVector2(65f, 40f);
        private static readonly MyVector2 TallPosition =
                new MyVector2(40f, 55f);
        private static readonly MyVector2 TestPosition =
                new MyVector2(45f, 45f);
        private static readonly MyVector2 FirstBigPosition =
                new MyVector2(55f, 60f);

        public static List<Position> AllPositions
        {
            get
            {
                // change to number of last position
                int maxPositionNumber = 0;
                List<Position> allPositions = new List<Position>();

                return allPositions;
            }
        }
    }
}
```

Výpis B.1. Připravená třída pro tvorbu nové linky

Je potřeba si na layoutu linky všimnout, že pozice mají rozlišné rozměry. Ty jsou v naší třídě definovány jako proměnné třídy $MyVector2$.

Proměnná $maxPositionNumber$ nám pomáhá číslovat pozice. Změňte její hodnotu na číslo poslední pozice, t.j. 70, jako na následujícím výpisu B.2.

```
int maxPositionNumber = 70;
```

Výpis B.2. Aktualizovaná proměnná $maxPositionNumber$.

Při vytvoření nové pozice její hodnotu snížíme o 1, případně pokud nějaká čísla pozic jsou přeskočena, příslušně upravíme její hodnotu.

Pod proměnnou allPositions vytvoříme poslední pozici linky 70 (viz výpis B.3):

```
Position lastPos = new CrosswayPosition(
    positionNumber: maxPositionNumber--,
    measures: SmallPosition,
    partOfLine: PartOfLine.Kenso,
    transitionPositionFrom: Direction.Left,
    transitionPositionTo: Direction.Right,
    allPositions: allPositions
);
```

Výpis B.3. Definice poslední pozice linky

Jak určují specifika pozic, jedná se o křižovatku, protože je to poslední pozice linky. Proměnnou $maxPositionNumber$ jsme po použití snížili o 1, $PartOfLine$ je enum definovaný v $Models/Utils/Enums.cs$ a protože pozice patří do části $Kenso$, použili jsme $PartOfLine.Kenso$. Dále je potřeba definovat, jakým směrem se má jednotka

pohybovat. Jedná se o *transitionPositionFrom* a *transitionPositionTo* typu enum *Direction*, který je definován ve výše zmíněné třídě *Enums.cs*. Ten je následně převeden v konstruktoru na typ *MyVector*2 pomocí slovníku *DirectionToMyVector*2*Dic* definovaném v *Models/Utils/DirectionToDict.cs*. U ostatních pozic se již směr pohybu jednotky definovat nemusí. Stačí definovat relativní směr k následující pozici a pohyb jednotky se vypočítá automaticky.

Pozici není potřeba přidávat do listu *allPositions*, její přidání je provedeno uvnitř konstruktoru.

Následují pozice 57-69. Jelikož jediný rozdíl mezi pozicemi je v čísle, usnadníme si jejich vytváření *for* smyčkou (výpis B.4).

```
for (int = 0; 26 > i; i++)
{
    Position KensoLinePos = new Position(
        positionNumber: maxPositionNumber,
        measures: SmallPosition,
        relativeLocationToNextPosition: Direction.Left,
        partOfLine: PartOfLine.Kenso,
        allPositions: allPositions
    );
    if (i % 2 == 1)
        maxPositionNumber--;
}
```

Výpis B.4. Definice pozic s číslem 57-69

Celkem se jedná o 26 pozic a každé číslo pozice se v tomhle úseku vyskytuje dvakrát. Proto se snížení *maxPositionNumber* děje v *if* podmínce, která proběhne každý druhý průběh. Proměnná *relativeLocationToNextPosition* definuje směr aktuální pozice vzhledem k následující pozici (pozici s větším číslem, vytvořenou před ní). Proměnná očekává typ enum *Direction*, v tomto případě *Direction.Left* (pozice 66 se nachází vlevo od pozice 57, proto *Direction.Left*).

Dále jsou pozice 53-56, které můžeme spárovat do další *for* smyčky, jako ve výpisu B.5.

```
for (int = 0; 2 > i; i++)
{
    Position Kenso55_56 = new Position(
        positionNumber: maxPositionNumber--,
        measures: SmallPosition,
        relativeLocationToNextPosition: Direction.Bottom,
        partOfLine: PartOfLine.Kenso,
        allPositions: allPositions
    );
}

for (int = 0; 2 > i; i++)
{
    Position Kenso54_53 = new Position(
        positionNumber: maxPositionNumber--,
        measures: WidePosition,
        relativeLocationToNextPosition: Direction.Right,
        partOfLine: PartOfLine.Kenso,
```

```
            allPositions: allPositions
        );
    }
```

Výpis B.5. Definice pozic s číslem 53-66

Pozice 53 a 54 jsou širší než ty, které jsme vytvořili do teď. Použijeme proto větší rozměr: *WidePosition*.

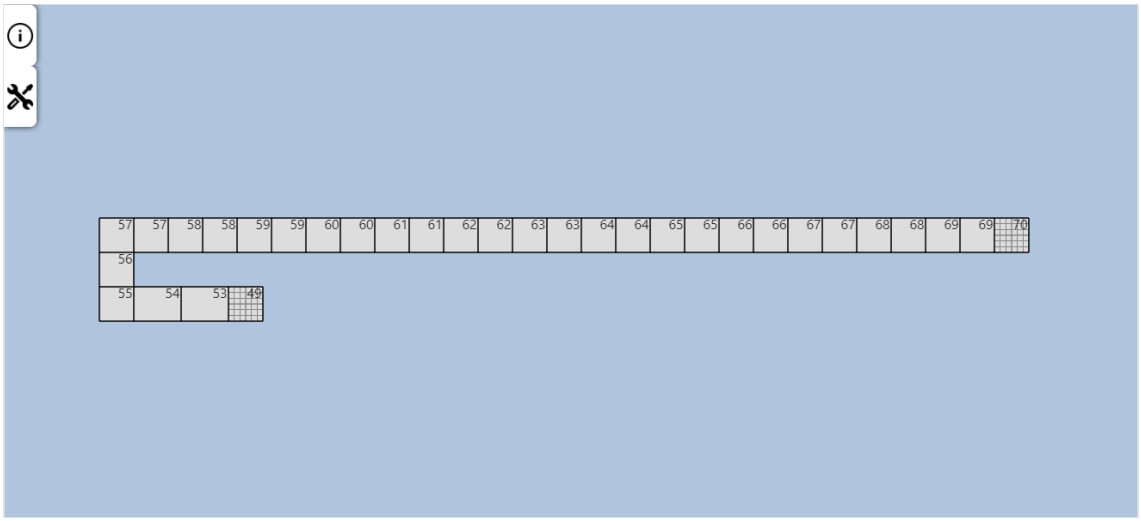
Další v pořadí je pozice 49. Jedná se o první pozici části *Kenso*, proto to musí být křižovatka. Zároveň jsme přeskočili 3 čísla pozic a musíme *maxPositionNumber* relevantně upravit. Pozice 50-51 nejsou pro nás relevantní (viz kapitola 18 - Úprava modelu). Definice pozice je vidět ve výpisu B.6.

```
maxPositionNumber -= 3;

Position Kenso49 = new CrosswayPosition(
        positionNumber: maxPositionNumber--,
        measures: SmallPosition,
        relativeLocationToNextPosition: Direction.Right,
        partOfLine: PartOfLine.Kenso,
        allPositions: allPositions
    );
```

Výpis B.6. Definice pozice s číslem 49

Tímto je část *Kenso* dokončena a měla by vypadat jako na obrázku B.2.



Obrázek B.2. Vymodelovaná část *Kenso* linky R5

Nyní přidáme část testů *Kimitsu*1. Podle specifik musí být po pozicích testů (tj. 32-47) poslední testovací pozice. V tomto případě bude mít číslo 48. Její vytvoření bude vypadat následovně (výpis B.7):

Následují dvojice testovacích pozic a testovacích křižovatek. Jako první se vytvoří testovací křižovatka, pomocí které se následně vytvoří samotná testovací pozice.

```
for(int i = 0; 8 > i; i++)
{
    TestCross Kimitsu1Test = new TestCross(
        positionNumber: maxPositionNumber--,
```

```
Position lastTest48 = new LastTestPosition(
    positionNumber: maxPositionNumber--,
    measures: SuperWidePosition,
    relativeLocationToNextPosition: Direction.Right,
    partOfLine: PartOfLine.Kimitsu1,
    allPositions: allPositions
);
```

Výpis B.7. Definice poslední testovací pozice s číslem 48

```
        measures: WidePosition,
        relativeLocationToNextPosition: Direction.Right,
        partOfLine: PartOfLine.Kimitsu1,
        allPositions: allPositions
    );

    Kimitsu1Test.CreateTestPosition(
        measures: TestPosition,
        allPositions: allPositions,
        relativeLocationToPosition: Direction.Bottom);
    maxPositionNumber--;
}
```

Výpis B.8. Definice testovacích stanic s čísly 32-47

Ve výpisu B.8 byla testovací pozice vytvořena pomocí metody testovací křižovatky *CreateTestPosition*, kde je nutno specifikovat relativní pozici příslušného testu. Tady bylo použito *Direction.Bottom*. Nesmí se zapomenout na snížení *maxPositionNumber*.

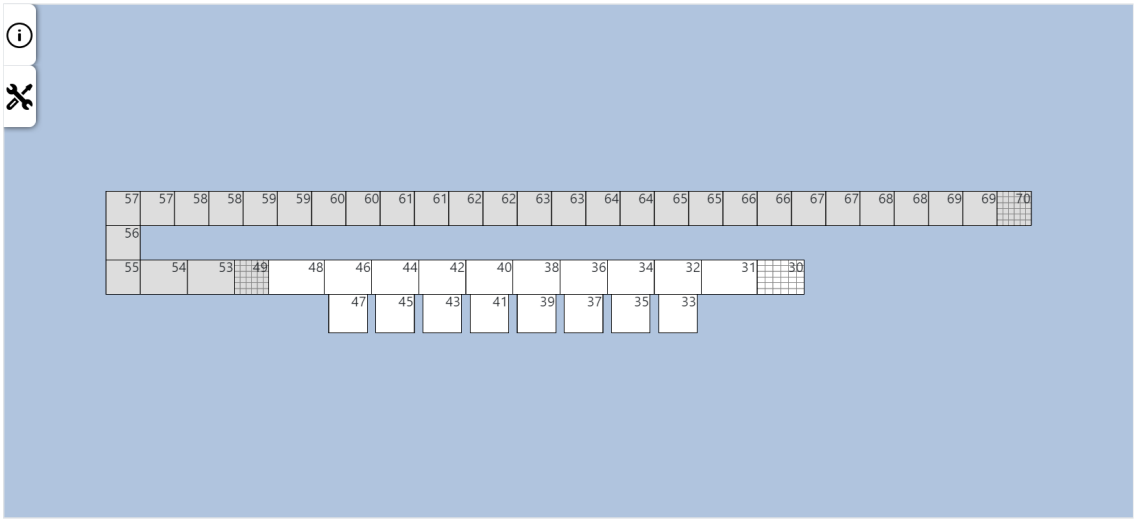
Na závěr chybí první testovací pozice, která je přilehlá testům, a nakonec křižovatka jako první pozice našeho úseku tvořené linky (výpis B.9).

```
Position firstTest31 = new FirstTestPosition(
    positionNumber: maxPositionNumber--,
    measures: SuperWidePosition,
    relativeLocationToNextPosition: Direction.Right,
    partOfLine: PartOfLine.Kimitsu1,
    allPositions: allPositions
);

Position Kimitsu130 = new CrosswayPosition(
    positionNumber: maxPositionNumber--,
    measures: WidePosition,
    relativeLocationToNextPosition: Direction.Right,
    partOfLine: PartOfLine.Kimitsu1,
    allPositions: allPositions
);
```

Výpis B.9. První testovací pozice a křižovatka na začátku části *Kimitsu*1

Tímto je námi vybraný úsek linky hotový a měl by vypadat následovně (obrázek B.3):



Obrázek B.3. Vymodelované části *Kenso* a *Kimitsu*1 linky R5

Pokud však chceme, aby se nám na lince správně zobrazovaly jednotky, musíme určit, s jakými částmi linky má aplikace pracovat. To se provede v třídě *Models/UpdateDataManager.cs* ve funkci *UpdateDataFromDB* (výpis B.10).

```
private void UpdateDataFromDB()
{
    UpdateUnits(
        DbDataAccess.GetUnitsFrom⟨PartKaishi⟩(PartOfLine.Kaishi));
    UpdateUnits(
        DbDataAccess.GetUnitsFrom⟨PartKimitsu1⟩(PartOfLine.Kimitsu1));
    UpdateUnits(
        DbDataAccess.GetUnitsFrom⟨PartKenso⟩(PartOfLine.Kenso));
    UpdateUnits(
        DbDataAccess.GetUnitsFrom⟨PartKimitsu2⟩(PartOfLine.Kimitsu2));
    UpdateUnits(
        DbDataAccess.GetUnitsFrom⟨PartUnten⟩(PartOfLine.Unten));
    UpdateUnits(
        DbDataAccess.GetUnitsFrom⟨PartKansei⟩(PartOfLine.Kansei));
    UpdateUnits(
        DbDataAccess.GetUnitsFrom⟨PartKonpo⟩(PartOfLine.Konpo));

    // finished units
    UpdateFinishedUnits(
        DbDataAccess.GetUnitsFrom⟨PartMol⟩(PartOfLine.Finished));
    // NG units
    UpdateUnits(DbDataAccess.GetUnitsFromRepairs());
}
```

Výpis B.10. Funkce *UpdateDataFromDB* s aktualizací dat ze všech částí linky

V této funkci je zapotřebí zakomentovat nepoužívané části, případně přidat ty, co tam chybí. V našem případě se jedná o části *Kimitsu*1 a *Kenso*, kód by proto měl vypadat jako na výpisu B.11.

```
private void UpdateDataFromDB()
{
    UpdateUnits(
        DbDataAccess.GetUnitsFrom⟨PartKimitsu1⟩(PartOfLine.Kimitsu1));
    UpdateUnits(
        DbDataAccess.GetUnitsFrom⟨PartKenso⟩(PartOfLine.Kenso));

    // finished units
    UpdateFinishedUnits(
        DbDataAccess.GetUnitsFrom⟨PartMol⟩(PartOfLine.Finished));
    // NG units
    UpdateUnits(DbDataAccess.GetUnitsFromRepairs());
}
```

Výpis B.11. Funkce *UpdateDataFromDB* po úpravě, pouze s potřebnými částmi

B.2.2 Změna konstant

V souboru *Models/Constants.cs* je možné provést úpravy aplikace. Jedná se o následující možnosti:

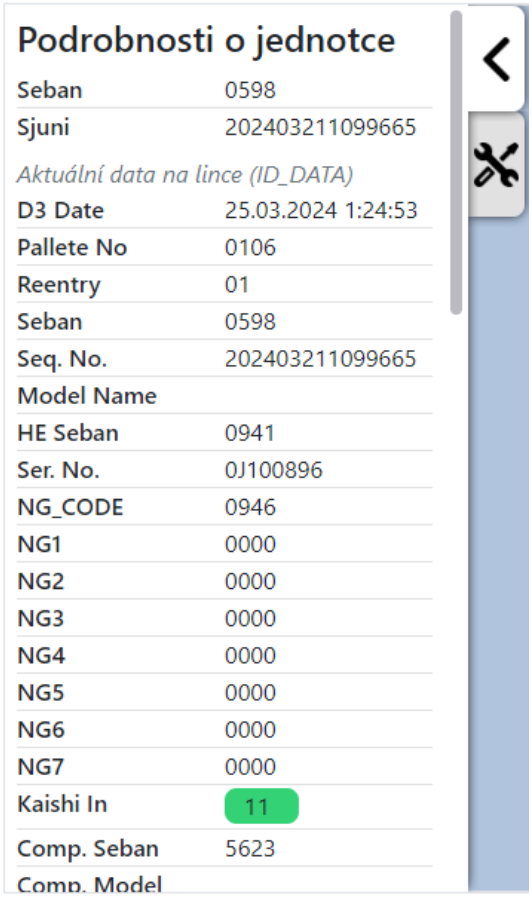
- **Frekvence updatu dat** (*UpdateInterval*) - Frekvence tiku. Vyhodnocování nové pozice jednotek je prováděno v pravidelné smyčce, která se vykoná každých x milisekund.
- **Frekvence dotazů na databázi** (*DBAccessTickCounter*) - Není potřeba dotazovat se databáze každý tik. Tato hodnota určuje, kolik tiků má proběhnout, než se získají nová data z databáze.
- **Rychlost pohybu jednotky** (*UnitSlowSpeed*) - Dráha pozice se bere jako jedna jednotka. Tato hodnota udává, o jaký kus dráhy se má jednotka za jeden tik po pozici posunout.
- **Rychlost pohybu rychlejší jednotky** (*UnitFastSpeed*) - Jednotka, která je zpožděná oproti skutečné poloze, má zvýšenou rychlost, aby zpoždění co nejdříve vyrovnala.
- **Rychlost mizení/objevování se jednotky** (*AnimationDelta*) - Určuje, o kolik se změní průhlednost jednotky za jeden tik.
- **Aktuálně používaný layout** (*ActualLayout*) - Layout, který je aplikací používán.

B.2.3 Změna obrázků jednotek

Pokud je potřeba změnit zobrazované obrázky modelů, či upravit, pro jaké jednotky jsou přiřazeny jaké obrázky modelů, je možné tak provést v souboru *wwwroot/js/Graphics/utilities/SVGSetUp.js*. Zde se nachází slovník *SVGDict*, kde je možné provést žádané úpravy. Klíč slovníku určuje název SVG souboru, který se použije pro model jednotky. Tento soubor se musí nacházet ve složce *wwwroot/textures*. Hodnota klíče je seznam *Seban*ů jednotek.

B.2.4 Výběr zobrazovaných podrobností o jednotce

Podrobnosti o jednotce jsou zobrazeny po kliku myši (obrázek B.4). Kromě *Sebanu* a *Sjuni*, které jsou identifikátory jednotky, se také získají podrobnější data z databáze. Konkrétně se jedná o dvě tabulky *IdData* a *Hinshitsu*.



Obrázek B.4. Zobrazení podrobností o jednotce

Data z tabulky se zobrazí pouze pokud se v nich záznamy jednotek nacházeli. To, které údaje se z tabulek zobrazí, je možné nastavit v třídě *Models/Database/UnitDBInfoStruct.cs*. Zde se nacházejí dvě proměnné, každá pro jednu tabulku, o typu *List*⟨InfoStruct⟩. Definice pro zobrazení řádku v podrobnostech může vypadat následovně (výpis B.12):

```
new InfoStruct(colName: "RSTKaishiIn", des: "Kaishi In", isRst: true)
```

Výpis B.12. Použití třídy *InfoStruct*

Proměnná *colName* definuje název sloupce z tabulky, *des* určuje, pod jakým popisem se hodnota zobrazí a *isRst* barevně formátuje hodnotu (např. hodnota *KaishiIn* na obrázku B.4).

Příloha C
Otázky z uživatelského testování

- Ukaž v aplikaci na úsek linky *Kansei*.
- Je grafické rozdělení částí jednoznačné/dostatečné?
- Najdi sériové číslo jednotky, která se testuje na pozici 105.
- Kolik jednotek se nachází v opravě?
- Proč jsou pozice 30, 49, 91, 119, 143, 180 odlišné?
- Co znamená mizení jednotek na pozici 49?
- Jsou obrázky jednotek dostatečné?
- Je něco, co by si na aplikaci změnil/přidal/odebral?

Příloha D
Projekt

Adresář s přiloženými soubory, stejně jako repozitář GitLabu: `https://gitlab.fel.cvut.cz/kaslonat/monitoring_bp`, obsahuje dva projekty:

- OnlineMonitoring - webová aplikace vizualizace
- ManageDB - aplikace na ovládání databáze