VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Aplikovaná informatika

VYTVOŘENÍ INTERAKTIVNÍCH MODELŮ HISTORICKÉHO SLADOVNICKÉHO VYBAVENÍ

Bakalářská práce

Autor práce: Zlata Valakhanovich

Vedoucí práce: PaedDr. František Smrčka, Ph.D.

Jihlava 2026

Vysoká škola polytechnická Jihlava

Tolstého 16, 586 01 Jihlava

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Autor práce:	**Zlata Valakhanovich**
Studijní program:	Aplikovaná informatika
Garant studijního programu:	doc. Ing. Lenka Kuklišová Pavelková, Ph.D.
Název práce:	**Vytvoření interaktivních modelů historického sladovnického technologického vybavení**
Vedoucí práce:	PaedDr. František Smrčka, Ph.D.
Cíl práce:	Cílem práce je vytvoření webové stránky s interaktivními modely historického sladovnického technologického vybavení, která bude sloužit k propagaci kulturní památky, bývalého pivovaru na tvrzi Starý hrad v Želetavě. Modely budou vytvořeny na základě historických skic. Pro realizaci budou využity grafické programy Affinity pro počítač a Affinity Designer pro tablet, dále skriptovací jazyk JavaScript a knihovna GSAP. Jednotlivé modely budou opatřeny animacemi, přičemž první z nich bude obsahovat vyskakovací okno (pop-up window). Webová stránka bude navržena responzivně tak, aby byla plně funkční i na mobilních zařízeních.

Abstrakt

Bakalářská práce se zabývá možnostmi webové vizualizace historického vývoje sladovnické výroby v Želetavě s důrazem na výkon a optimalizaci pro mobilní zařízení. Teoretická část se zaměřuje na historii místní sladovny a na popis technologického procesu výroby sladu. Následně práce analyzuje dostupné webové animační technologie a porovnává přístupy založené na CSS a JavaScriptu, se zvláštním důrazem na SVG grafiku a animační knihovnu GSAP. Pozornost je věnována vlivu animací na výkon, spotřebu systémových prostředků a uživatelský komfort na desktopových i mobilních zařízeních. Cílem práce je navrhnout koncept vizuálně kvalitních a technicky efektivních SVG animací, které budou v budoucnu sloužit jako základ interaktivní webové prezentace.

Klíčová slova

sladovnictví; webová vizualizace; SVG grafika; webové animace; GSAP; interaktivní vizualizace

Abstract

Bachelor's thesis deals with the possibilities of the historical development of malting production in Želetava visualization, with an emphasis on performance and optimization for mobile devices. The theoretical part focuses on the history of the local malt house and on a description of the technological process of malt production. Subsequently, the thesis analyzes available web animation technologies and compares CSS and JavaScript-based approaches, with particular attention paid to SVG graphics and the GSAP animation library. Special consideration is given to the impact of animations on performance, system resource consumption, and user experience on both desktop and mobile devices. The aim of the thesis is to propose a concept of visually refined and technically efficient SVG animations that can serve as a foundation for a future interactive web presentation.

Keywords

malting; web visualization; SVG graphics; web animations; GSAP; interactive visualization

Prohlašuji, že předložená bakalářská práce je původní a zpracoval/a jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil/a autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „**AZ**“).

Byl/a jsem seznámen/a s tím, že na mou bakalářskou práci se plně vztahuje **AZ**, zejména § 60 (školní dílo).

Podle § 47b zákona o vysokých školách souhlasím se zveřejněním své práce podle Směrnice pro vedení, vypracování a zveřejňování závěrečných prací na VŠPJ, a to bez ohledu na výsledek obhajoby.

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom/a toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mě požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutím licence.

V Jihlavě dne 7. dubna 2026

…….

Podpis studentky

Poděkování

Ráda bych touto cestou poděkovala své rodině, a především svým rodičům, za jejich neustálou podporu, trpělivost a zázemí, které mi vytvářeli po celou dobu mého studia. Bez vaší pomoci a důvěry bych tohoto cíle dosáhla jen stěží.

Obsah

Seznam obrázků

Seznam zkratek

API	Application Programming Interface (programové rozhraní)
ARIA	Accessible Rich Internet Applications (standard pro přístupnost webu)
CDN	Content Delivery Network (síť pro doručování obsahu)
CSS	Cascading Style Sheets (kaskádové styly)
DOM	Document Object Model (model objektů dokumentu)
FPS	Frames Per Second (snímky za sekundu)
GSAP	GreenSock Animation Platform (JavaScriptová animační knihovna)
HTML	HyperText Markup Language (jazyk pro tvorbu webových stránek)
PC	Personal Computer
SVG	Scalable Vector Graphics (škálovatelná vektorová grafika)
WebGL	Web Graphics Library (technologie pro 3D grafiku v prohlížeči)

Úvod

Bakalářská práce se zabývá tvorbou webových stránek s interaktivními modely historického sladovnického technologického vybavení. Cílem práce je vytvořit webovou aplikaci, která umožní návštěvníkům seznámit se s technologiemi dříve používanými při výrobě sladu. Projekt má kulturně-historický záměr, neboť přispívá k uchování a popularizaci technického dědictví spojeného s tradičními výrobními postupy a sladovnickými zařízeními, která jsou dnes často známa pouze odborníkům.

Impulsem k volbě tématu byla snaha propojit moderní webové technologie s historickým obsahem a podpořit tak zájem veřejnosti o regionální historii. Výsledná webová stránka bude součástí prezentace historického areálu ve Starém Hradci a umožní uživatelům prostřednictvím interaktivních modelů poznat, jak v minulosti fungovala sladovnická výroba. Téma je relevantní nejen z hlediska technologického zpracování, ale i společenského přínosu. Projekt může sloužit jako vzdělávací a propagační nástroj pro návštěvníky regionu Vysočina a širší veřejnost.

Práce je zaměřena především na praktickou implementaci. Bude zde popsán celý proces tvorby interaktivních modelů, počínaje převodem původních technických výkresu do digitální podoby, přes návrh animací až po jejich integraci do webového rozhraní. Výsledné modely budou vytvořeny ve formátu SVG a animovány pomocí skriptovacího jazyka JavaScript a vybrané animační knihovny GSAP. Důraz bude kladen na optimalizaci pro mobilní zařízení, aby byla aplikace snadno přístupná širokému okruhu uživatelů.

Při řešení úkolu budou použity metody analýzy požadavků, návrhu uživatelského rozhraní, implementace webové animace a testování funkčnosti. Alternativně zvažované přístupy, jako například využití čistě video formátu, byly zamítnuty z důvodu ztráty interaktivity a omezených možností přizpůsobení uživatelským akcím. Zvolený přístup umožní vytvořit dynamickou prezentaci, která uživatelům poskytne aktivní zážitek z poznávání historických technologií.

Přínosem práce bude vytvoření funkční webové stránky s kompletním souborem animovaných modelů, které mohou být využity při propagaci a prezentaci kulturního dědictví. Projekt přispěje ke zvýšení povědomí o historii sladovnictví a zároveň ukáže, jak lze moderní webové technologie uplatnit v oblasti popularizace technických a historických témat.

1 Teoretická část

Následující teoretická část práce slouží jako nezbytný základ pro pochopení kontextu praktické realizace. Její obsah je rozdělen do dvou logických celků: text nejprve definuje klíčové fáze sladovnického procesu, které budou předmětem animace, a následně poskytuje kritické srovnání grafických formátů a animačních knihoven, které vedlo k výběru technologie postavené na formátu SVG a knihovně GSAP.

1.1 Historie sladovnické výroby v Želetavě

Kapitola se zaměřuje na historický vývoj sladovnické výroby v Želetavě v širším hospodářském a technologickém kontextu. Popisuje vznik a rozvoj sladovny jako součásti místního pivovarnického areálu, její význam v regionu a postupné změny ve využití objektu. Součástí kapitoly je rovněž přiblížení technologického procesu výroby sladu a jeho proměn v průběhu času.

1.1.1 Historie sladovny v Želetavě

Lze předpokládat, že objekty Starého hradu v Želetavě byly vybudovány pravděpodobně v 16. století s pomocí pánů z Hradce. První písemná zmínka pochází z roku 1568, kdy Zachariáš z Hradce svou telčskou mocí podpořil výstavbu rybníků, sladovny, pivovaru a jiných budov v Želetavě. Na konci století již pivovar dodával pivo do místních hostinců (Doležal a kol., 2015).

Významným milníkem v historii byl rok 1862, kdy se novým majitelem stal Carl Friedrich Kammel von Hardegger. Z jeho iniciativy byly založeny nové průmyslové podniky, včetně nového parostrojního pivovaru. Původní želetavský pivovar (Starý hrad) byl od té doby využíván pouze pro sladovnické účely (Doležal a kol., 2015).

V roce 1868 byla sladovna ve Starém hradě zrekonstruována a vybavena jedním z nejmodernějších vertikálních vzdušných hvozdů anglického typu od firmy Noback & Fritze v Praze. Až do roku 1901, kdy Carl Kammel zemřel, se pivovar neustále rozvíjel: bylo zavedeno telefonní vedení, automatické smolení sudů a proběhla rekonstrukce chladírny (Doležal a kol., 2015).

S nástupem nové majitelky, dcery Carla, Anny Attems-Heiligenkreuz, však začal postupný úpadek. První rána přišla s první světovou válkou, po níž se již nikdy nepodařilo dosáhnout předválečného objemu produkce. Velkou konkurencí byl také Znojemský pivovar, který měl v blízkosti svůj sklad (Doležal a kol., 2015).

V roce 1925 byl pivovar elektrifikován, sladovna však nikoliv. Vlastní produkce sladu tak pro potřeby pivovaru nestačila a musel být dokupován slad z Plzně. V důsledku požáru v roce 1932 se v roce 1935 stal nájemcem pivovaru významný pivovarník Jan Staller. V únoru 1936 byla uzavřena kartelová dohoda, která platila až do roku 1944, tedy do období druhé světové války. Do dohody se zapojily prakticky všechny pivovary kromě několika podniků v okolí Želetavy, jako byl Vladislav, Brtnice atd (Doležal a kol., 2015).

S novým nájemcem přišel nový rozkvět pivovaru, který byl však ukončen druhou světovou válkou. Pivovar následně nebyl kvůli své lokalitě uznán jako perspektivní. Budova několikrát

změnila svůj účel – fungovala jako sýrárna a později sloužila pouze jako sklad (Doležal a kol., 2015).

1.2.1 Technologický proces výroby sladu

Výroba sladu se výrazně proměnila v období mezi předindustriální érou a 19. století. Dříve byly sladovna a pivovar vnímány jako jeden technologický celek, kde se připravil slad a následně uvařilo pivo. Díky technologickému pokroku se však výroba sladu mohla oddělit od samotného vaření piva, což umožnilo zásobování různých hostinců z centrálních sladoven. Často se stávalo, že pro přípravu sladu byly využívány budovy v hradních komplexech. Příkladem může být Plzeň, kde můžeme v současné době vidět pozůstatky bývalých prostor sladovny, jež byly částečně přeměněny na muzeum (Anderle, 2013).

Slad je uměle naklíčené obilné zrno. Cílem klíčení je narušení obalu zrna a aktivace enzymů, které v pozdější fázi výroby umožňují štěpení škrobů na zkvasitelné cukry (Anderle, 2013).

V minulosti probíhalo klíčení sladu jednoduše na podlaze tzv. humnech, kde se zrno převracelo ručně. V Želetavě však bylo instalováno zařízení se speciálními lopatkami různých velikostí umístěnými na ose válce, které zajišťovaly rovnoměrné obracení zrna (Doležal a kol., 2015). V současné době byl mechanismus zrestaurován a může být spuštěn pro muzeální účely.

Proces výroby sladu se dělí na následující fáze (The Processes of Malting Through the Ages, 1997):

1. Skladování a čištění ječmene: Ječmen byl očištěn od polních nečistot a skladován ve speciálních sýpkách. Bylo klíčové udržovat nízkou vlhkost, aby nedošlo ke znehodnocení zrna.
2. Máčení: Zrno, nejčastěji ječmen, se máčelo v cisternách ve vodě o teplotě okolo 12 °C. Doba máčení se v průběhu historie měnila; v 18. století se zkrátila na 72 hodin. Voda se několikrát měnila a zrnu se dopřávaly tzv. vzdušné přestávky, přibližně 8 hodin, před dalším namočením.
3. Odležení: Následně bylo zrno vysypáno ve vyšší vrstvě, aby se zahřálo. Například v Anglii byla procedura povinná z důvodu daně ze sladu, právě v moment kontroloři měřili objem vyrobeného sladu. Zákon platil do roku 1827. V současnosti procedura v moderních sladovnách chybí z důvodu odlišné konstrukce máčecích nádob.
4. Klíčení: Dále byl slad rozprostřen na podlahu humna k samotnému klíčení. Doba klíčení se postupně zkracovala, až byla ve druhé polovině 19. století ustálena na cca 14 dnech, což trvalo minimálně do 30. let 20. století.
5. Zavadání: Před přesunem do hvozdu se slad nechal mírně proschnout.
6. Hvozdění: Když tzv. zelený slad dosáhl požadovaného stupně prokvašení, byl přesunut na hvozd. Cílem bylo zastavit vegetační procesy a snížit vlhkost. Právě procedura dodává sladu jeho specifickou chuť a barvu.
7. Odklíčení a skladování: Po ukončení hvozdění byl slad zbaven kořínků, tzv. sladového květu, aby zbylo pouze zrno. Aby se zabránilo opětovnému navlhnutí sladu, byl skladován v dřevěných zásobnících či sudech, které pomáhaly regulovat vlhkost.

1.2 Technologie pro webovou vizualizaci

Cílem kapitoly je analyzovat a zdůvodnit výběr technologií použitých pro realizaci webové vizualizace v rámci práce. Text se zaměřuje na přehled dostupných přístupů a jejich porovnání z hlediska výkonu, interaktivity, výpočetní náročnosti a vhodnosti pro moderní webová a mobilní zařízení. Zvláštní pozornost je věnována technologii SVG a jejímu využití v kombinaci s JavaScriptovými animačními knihovnami, přičemž jsou srovnávány také alternativy, jako jsou Canvas, video a 3D grafika, s cílem identifikovat nejvhodnější řešení pro práci s komplexními grafickými scénami a animacemi.

1.2.1 Vektorová a rastrová grafika na webu

Volba grafického formátu je jedním z prvních aspektů, které je nutné zvážit před zahájením práce s animací. Obecně je známý problém při zvětšování rastrových obrázků, kdy dochází ke ztrátě kvality a jemné detaily se stávají nečitelnými. V oblasti webové grafiky se používá několik základních grafických formátů, mezi které patří například (Shapiro, 2015):

1. JPEG (Joint Photographic Experts Group): využíván především pro fotografický obsah
2. GIF (Graphics Interchange Format): používán zejména pro jednoduché rastrové animace
3. PNG (Portable Network Graphics): často využíván v grafickém designu pro ikony a obrázky s průhledným pozadím
4. SVG (Scalable Vector Graphics): určen pro škálovatelnou vektorovou grafiku, jako jsou ikony, loga nebo ilustrace

Existují také grafické formáty určené výhradně pro práci v konkrétní aplikaci, například formát PSD používaný v programu Adobe Photoshop. S formáty je však mimo dané softwarové prostředí obtížné nebo prakticky nemožné pracovat. Pro tvorbu animovaných a interaktivních prvků na webových stránkách je nejvhodnějším formátem SVG, a zejména díky jeho XML struktuře, která umožňuje přímou spolupráci s jazyky JavaScript a CSS (Shapiro, 2015).

SVG byl představen v roce 1999 a je založen na jazyce XML. Jedná se o značkovací jazyk určený pro popis grafiky, podobně jako HTML slouží k popisu textového obsahu. SVG je velmi flexibilní formát lze jej stylovat pomocí CSS, vkládat přímo do HTML dokumentu a dále s ním programově manipulovat za účelem vytváření animací a interaktivních efektů (Lersen, 2018).

Důvodem, proč se SVG dlouhou dobu nepoužívalo v širším měřítku, byla omezená podpora ve starších webových prohlížečích, zejména v Internet Exploreru, který SVG formát původně nepodporoval. Práce se SVG byla proto značně komplikovaná až do rozšíření JavaScriptových knihoven, jako je například Raphaël, které usnadnily jeho využití (Lersen, 2018).

Díky své povaze dokumentu je SVG součástí DOM, což umožňuje pohodlnou manipulaci s jednotlivými grafickými prvky, jejich animaci a reakci na uživatelské akce. Každý prvek může být adresován samostatně, například pomocí atributu id. Nespornou výhodou SVG je také zachování kvality obrazu při libovolném zvětšení, nezávisle na rozlišení nebo velikosti obrazovky (Shapiro, 2015).

Jednoduché grafické tvary, jako jsou čáry, obdélníky nebo elipsy, lze definovat přímo v kódu pomocí SVG značek. V případě složitějších ilustrací je však SVG nejčastěji generováno pomocí grafických editorů, jako jsou Inkscape, Affinity Designer nebo Adobe Illustrator. Nevýhodou

nástrojů je skutečnost, že jsou primárně navrženy pro potřeby grafiků a ilustrátorů, nikoli pro optimalizaci SVG souborů pro použití na webu (Lersen, 2018).

1.2.2 Možnosti animace: CSS vs. JavaScript

V některých případech je z hlediska uživatelského rozhraní (UI) a uživatelské zkušenosti (UX) výhodné nahradit animaci implementovanou v jazyce JavaScript animací realizovanou pomocí CSS. Jakmile však animace přestane být elementární a její průběh se stane složitějším, dochází k výraznému nárůstu komplexity kódu, snížení jeho čitelnosti a obtížnější údržbě (Shapiro, 2015).

CSS se snaží řešit časovou posloupnost animací pomocí tzv. keyframes, které umožňují rozdělit časovou osu animace na jednotlivé úseky s definovanými změnami vlastností. Nevýhodou tohoto přístupu je skutečnost, že časování je zpravidla vyjádřeno v procentech. Jakákoliv změna v konkrétním časovém okamžiku (například v jedné sekundě animace) pak vyžaduje přepočítání všech procentuálních poměrů v rámci celé animace, což snižuje flexibilitu řešení (Lersen, 2018).

Existují však situace, ve kterých je použití CSS animací nejen dostačující, ale i doporučované. Typickým příkladem je jednoduchá změna barvy nebo jiného vizuálního atributu prvku při najetí kurzorem myši (hover efekt).

Volba mezi CSS a JavaScript animacemi úzce souvisí s principy kvalitního kódu. Kvalitní kód je především čitelný, stručný a snadno rozšiřitelný. Čím menší je objem kódu při zachování funkčnosti, tím jednodušší je jeho dlouhodobá údržba a opětovné použití v budoucích projektech. Poněvadž se obecně nedoporučuje implementovat CSS animace u prvků, u nichž se v budoucnu předpokládá animace řízená pomocí JavaScriptu (Shapiro, 2015).

CSS animace jsou ve své podstatě lineární: vývojář definuje změny jednotlivých vlastností (properties) pomocí pravidel, která určují, jak se prvek v čase vykresluje. Dlouhou dobu byl však problémem omezený rozsah vlastností SVG, které bylo možné pomocí CSS animovat, což jejich využití v komplexnějších scénářích značně limitovalo (Shapiro, 2015).

Zde je CSS animace SVG prvku, změna vzhledu tlačítka při kliknutí. Doporučuje se uvést více variant, například animaci vlastnosti fill a samostatně animaci pomocí transform: translate(), a následně příklady slovně popsat s uvedením názvů jednotlivých prvků (Lersen, 2018).

Změna barvy (fill) při interakci s tlačítkem:

```css
#icon {
    fill: #555;
    transition: fill 0.3s ease;
}

button:active #icon {
    fill: #e63946;
}
```

Obr. 1: Změna barvy

Zdroj: vlastní zpracování dle Lersen (2018)

Transformace SVG prvku pomocí posunu (translate):

```css
#icon {
    transition: transform 0.4s ease;
}

button:hover #icon {
    transform: translateX(10px);
}
```

Obr. 2: Transformace SVG prvku

Zdroj: vlastní zpracování dle Lersen (2018)

V obou případech je animace řízena změnou CSS vlastností (fill, transform), přičemž SVG prvek je identifikován pomocí atributu id.

Nyní se zaměříme na animace implementované pomocí čistého JavaScriptu. V minulosti bylo vytváření animací v JavaScriptu problematické, protože vyžadovalo ruční implementaci smyček a optimalizaci výkonu, zejména z hlediska počtu snímků za sekundu (FPS). S postupem času však začaly vznikat knihovny, jako například jQuery, které abstrahovaly složitou logiku a umožnily vnímat animace jako součást aplikačního rozhraní (Lersen, 2018).

Navzdory existenci velkého množství knihoven má čistý JavaScript i nadále své opodstatnění a zůstává silným nástrojem pro tvorbu animací, zejména v případech, kdy je vyžadována maximální kontrola nad chováním aplikace (Shapiro, 2015).

Pro zvýšení interaktivity animací lze využít metodu addEventListener, která umožňuje reagovat na různé uživatelské události. K jednomu prvku je možné připojit více obslužných funkcí (handlers), což je výhodné jak při práci s čistým JavaScriptem, tak při integraci s knihovnami (Lersen, 2018).

Mechanismus funguje tak, že je k danému prvku přiřazena funkce nebo objekt, který implementuje metodu handleEvent(). Metoda je následně volána v okamžiku, kdy na cílovém prvku (event target) dojde k definované události (Lersen, 2018).

```javascript
const button = document.getElementById("animateBtn");
const icon = document.getElementById("icon");

button.addEventListener("click", () => {
    icon.style.transform = "scale(1.2)";
});
```

Obr. 3: Základní použití EventListener

Zdroj: vlastní zpracování dle Lersen (2018)

Přístup umožňuje vytvářet komplexní, dynamické a kontextově závislé animace, které by bylo pomocí samotného CSS obtížné nebo nemožné realizovat. Je nutné postupovat obezřetně při opakovaném používání obslužných funkcí událostí. Může nastat situace, kdy je k určitému cílovému prvku nejprve připojena anonymní funkce prostřednictvím metody addEventListener a později je ke stejnému prvku přidána další, identická anonymní funkce. V takovém případě jsou obě funkce uloženy v seznamu posluchačů událostí a budou vykonávány současně. Přístup může vést k nežádoucímu chování aplikace, například k vícenásobnému spouštění animací, a v dlouhodobém horizontu také k problémům se správou paměti. Poněvadž je vhodné používat pojmenované funkce, které lze v případě potřeby odebrat pomocí metody removeEventListener (MDN, 2026).

Příklad správného použití posluchače událostí je uveden níže. Animace je zde spuštěna přidáním třídy animated, která reprezentuje animovaný stav prvku a umožňuje jasně oddělit statickou a dynamickou podobu komponenty (MDN, 2026).

```
function animateIcon() {
    const icon = document.getElementById("icon");
    icon.classList.add("animated");
}

const button = document.getElementById("animateBtn");
button.addEventListener("click", animateIcon);
```

Obr. 4: Ukázka animated

Zdroj: vlastní zpracování dle MDN (2025)

Použití samostatné třídy pro animovaný stav zvyšuje čitelnost kódu a usnadňuje jeho další rozšiřování. Přístup je vhodnější než přímá manipulace s jednotlivými CSS vlastnostmi přímo v JavaScriptu (MDN, 2026).

Z uvedených důvodů je v rámci projektu nejvhodnější propojit práci s formátem SVG se specializovanou animační knihovnou. Jednou z nejrozšířenějších a nejvýkonnějších knihoven v této oblasti je GreenSock Animation Platform, zkráceně GSAP. Jedná se o moderní nástroj, který umožňuje návrhářům i front-end vývojářům vytvářet animace založené na časové ose s výrazně přesnější kontrolou nad průběhem animace než v případě CSS animací založených na @keyframes nebo vlastnosti animation. Knihovna je optimalizována pro vysoký výkon a zajišťuje konzistentní chování napříč webovými prohlížeči (FreeCodeCamp, 2025).

GSAP umožňuje vytvářet interaktivní animace i s relativně základní znalostí jazyka JavaScript. Pro základní použití postačuje přidání knihovny prostřednictvím CDN odkazu do hlavičky HTML dokumentu, přičemž další pluginy lze připojit podle potřeb konkrétní implementace (FreeCodeCamp, 2025).

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/gsap/1.20.3/TweenMax.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/gsap@3.13.0/dist/gsap.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/gsap@3.13.0/dist/DrawSVGPlugin.min.js"></script>
```

Obr. 5 Přidání knihovny GSAP

Zdroj: vlastní zpracování dle FreeCodeCamp (2025)

Základním stavebním prvkem knihovny GSAP jsou tzv. tweens, které představují plynulý přechod mezi dvěma stavy vlastností objektu v čase. Pro vytvoření animace libovolného HTML nebo SVG prvku je nejprve nutné prvek identifikovat, následně definovat vlastnosti, které mají být animovány, a stanovit délku animace. Pro složitější animační sekvence nabízí GSAP práci s časovou osou pomocí objektu Timeline, jehož koncept je srovnatelný s vizuálními časovými osami používanými v nástrojích jako Adobe After Effects, avšak jeho implementace probíhá programově (FreeCodeCamp, 2025).

Následující ukázka demonstruje jednoduché použití objektů TweenLite a TimelineLite, které umožňují vytvořit sekvenční animaci SVG prvku identifikovaného pomocí atributu id (FreeCodeCamp, 2025).

```
var tl = new TimelineLite();

tl.from("#icon", 0.6, {
    scale: 0,
    opacity: 0
});

tl.to("#icon", 0.4, {
    rotation: 360
});
```

Obr. 6: Použití TimelineLite

Zdroj: vlastní zpracování dle FreeCodeCamp (2025)

Díky využití časové osy je možné jednotlivé animační kroky přehledně řetězit, upravovat jejich časování a snadno je rozšiřovat o další interakce nebo reakce na uživatelské události, což činí přístup vhodným řešením pro komplexnější webové projekty (FreeCodeCamp, 2025).

V další části práce se setkáváme s pluginem DrawSVG, který je součástí ekosystému knihovny GSAP a slouží k animaci kreslení SVG cest. Pro práci s pluginem je nutné mít předem připravený SVG soubor, který je exportován přímo do kódu dokumentu. Animace se následně realizuje nad jednotlivými elementy typu path, přičemž každý z nich je identifikován pomocí atributu id. Přístup umožňuje přesnou kontrolu nad průběhem animace jednotlivých částí ilustrace (FreeCodeCamp, 2025).

Princip fungování pluginu DrawSVG spočívá v manipulaci s vlastností délky SVG cesty, díky čemuž lze simulovat efekt postupného vykreslování objektu. Následující ukázka demonstruje základní použití pluginu při animaci SVG cesty (FreeCodeCamp, 2025).

```
gsap.fromTo("#linePath",
    { drawSVG: "0%" },
    { drawSVG: "100%", duration: 1.5 }
);
```

Obr. 7: Použití linePath

Zdroj: vlastní zpracování dle FreeCodeCamp (2025)

V uvedeném příkladu je SVG prvek s identifikátorem linePath animován od nulové délky po plnou délku cesty v definovaném časovém úseku. Animaci lze dále rozšířit o časovou osu, interakce s uživatelem nebo kombinaci s dalšími animačními vlastnostmi, jako je změna průhlednosti či transformace (FreeCodeCamp, 2025).

Na závěr je vhodné shrnout širší kontext využití JavaScriptových animací na webu. V praxi se lze setkat s problémem, kdy část vývojářské komunity nemá dostatečné povědomí o možnostech moderních JavaScriptových animačních nástrojů. Poněvadž je specialista často nucen přísně kontrolovat celý systém uživatelského rozhraní, omezovat použití animací s přesným časovým řízením, vyhýbat se fyzikálně založeným animacím a zároveň obětovat podporu starších webových prohlížečů (FreeCodeCamp, 2025).

Navzdory rozšířenému mýtu jsou JavaScriptové animace při správné implementaci stejně výkonné jako animace realizované pomocí CSS. Časté srovnávání CSS animací s JavaScriptovými animacemi však vychází z historického kontextu, kdy byly JavaScriptové animace reprezentovány především knihovnou jQuery. Moderní animační knihovny, jako je GSAP, však nabízejí výrazně vyšší úroveň optimalizace, přesnosti a kontroly nad animačním procesem než jQuery (Shapiro, 2015).

Použití CSS animací má i nadále své opodstatnění, zejména v případě jednoduchých efektů, jako jsou přechody při najetí kurzorem, kde CSS přechody přirozeně zapadají do existujících stylových souborů a nevyžadují další skriptovací logiku. Pro komplexnější, časově řízené a interaktivní animace je však JavaScriptová animace s využitím specializovaných knihoven efektivnějším a dlouhodobě udržitelným řešením (Shapiro, 2015).

1.2.3 Knihovny pro JavaScriptovou animaci a volba GSAP

V reálném prostředí webového vývoje se vývojáři každodenně setkávají s problémem kompatibility napříč webovými prohlížeči. Testování webových aplikací na tisících různých kombinací prohlížečů, verzí a zařízení je časově i finančně náročné. Existují sice specializované nástroje pro simulaci různých prostředí, avšak ony nikdy plně nenahradí reálné uživatelské chování. Dalším častým problémem je nekorektní struktura HTML nebo CSS kódu, například chybějící nebo neuzavřené značky, které mohou některé prohlížeče automaticky opravovat,

zatímco jiné nikoliv. V daných případech je doporučeno využívat validační nástroje, například W3C Validator (Shapiro, 2015).

Další komplikace mohou nastat při používání starších verzí prohlížečů, kde může docházet k degradaci animací nebo k jejich úplnému nezobrazení. Problematická může být rovněž kompatibilita rozvržení stránky, zejména pokud vývojář nepoužívá responzivní návrh založený na moderních technologiích. V takových případech se doporučuje využívat CSS Grid nebo Flexbox, které zajišťují konzistentní chování rozložení napříč různými prohlížeči a zařízeními (Lersen, 2018).

V dané souvislosti je vhodné zaměřit se na jednu z nejmodernějších JavaScriptových animačních knihoven, která problémy efektivně řeší, a to GreenSock Animation Platform (GSAP). Jedná se o výkonnou knihovnu určenou pro tvorbu vysoce optimalizovaných animací, která je snadno použitelná a dobře spolupracuje s technologiemi HTML, CSS, SVG i Canvas. Významnou výhodou je skutečnost, že základní verze knihovny je zdarma, je plně kompatibilní se všemi moderními prohlížeči a nabízí širokou škálu animovatelných vlastností, jako jsou transformace, barvy nebo průhlednost (GSAP SVG, 2025).

Základním principem knihovny GSAP je tzv. tweening, tedy proces plynulé animace mezi dvěma stavy objektu. Animace jsou vytvářeny pomocí metod gsap.to(), gsap.from() nebo gsap.fromTo(). Knihovna poskytuje zkrácené zápisy pro transformační vlastnosti, například x, y, rotation nebo scale, které jsou optimalizovány pro využití hardwarové akcelerace. Další konfigurační možnosti, jako jsou duration, ease, repeat, yoyo, stagger nebo události životního cyklu animace, umožňují velmi přesnou kontrolu nad chováním animací bez výrazného zvýšení složitosti kódu (GSAP SVG, 2025).

Pro objektivní zhodnocení výběru knihovny je vhodné provést srovnání s dalšími známými animačními knihovnami, například Anime.js nebo Popmotion. Moderní vývojáři se snaží vytvářet vizuálně atraktivní webové stránky, a díky popularitě JavaScriptu dnes existuje široká nabídka animačních nástrojů. Jedním z klíčových ukazatelů kvality animační knihovny je snímková frekvence (FPS), která přímo ovlivňuje plynulost animace. Při testování byly zohledněny scénáře s nízkou, střední i vysokou zátěží, přičemž byla sledována rychlost přepočtu stylů, překreslování prvků, práce s vrstvami a spotřeba paměti. Nadměrné využití paměti může vést ke zpomalení aplikace, zejména u složitějších animačních scénářů (Beňo a Ölvecký, 2024).

Při testech s tisícem animovaných uzlů dosahovaly všechny porovnávané knihovny přibližně 60 FPS. Při navýšení počtu uzlů z 1000 na 3000 však došlo k výrazným rozdílům ve výkonu. Knihovna Popmotion zaznamenala pokles snímkové frekvence z 60 FPS na přibližně 14,42 FPS, zatímco u GSAP došlo pouze k poklesu z 60 FPS na přibližně 37,03 FPS. Výsledky potvrzují, že GSAP je při vyšší zátěži výrazně výkonnější než konkurenční řešení (Vozisov a kol., 2019).

Při hodnocení výkonu animací je rovněž nutné brát v úvahu způsob, jakým webové prohlížeče zpracovávají vykreslování stránky. Proces vykreslovací pipeline se skládá z několika kroků, konkrétně z přepočtu stylů, aktualizace stromu vrstev, vykreslení obsahu a následného skládání vrstev. Efektivní animační knihovna minimalizuje zásahy do těchto kroků a tím zajišťuje plynulé vykreslování (Vozisov a kol., 2019).

Na základě provedených testů lze konstatovat, že GSAP patří mezi nejvýkonnější dostupné JavaScriptové animační knihovny. Další významnou výhodou je vlastní ekosystém pluginů a utilit,

který jiné knihovny v dané míře nenabízejí. GSAP rovněž poskytuje kvalitní podporu jak pro moderní, tak i starší webové prohlížeče. Ačkoli knihovna nebyla v roce 2019 tak rozšířená a na platformě GitHub neměla vysoký počet uživatelských hodnocení, do konce roku 2025 si získala výraznou popularitu a širokou komunitu uživatelů. Jednoznačnou výhodou použití GSAP je také jeho kompatibilita s dalšími technologiemi a frameworky, jako jsou jQuery, React a další běžně používané webové nástroje (FreeCodeCamp, 2025).

1.2.4 Výkon a optimalizace pro mobilní zařízení

Výkon aplikace ovlivňuje všechny aspekty projektu a jeho optimalizace má přímý dopad na výslednou kvalitu webového řešení. Řada odborných studií ukazuje, že nižší latence a rychlejší odezva webových stránek pozitivně ovlivňují jejich hodnocení ve výsledcích vyhledávačů. Animace představují pro výpočetní techniku relativně náročný proces a jejich souběžné používání může výrazně snížit výkon aplikace, což je nejčastěji patrné právě na mobilních zařízeních. V současné době je mobilní verze webu považována za prioritní, často důležitější než verze pro stolní počítače (Shapiro, 2015).

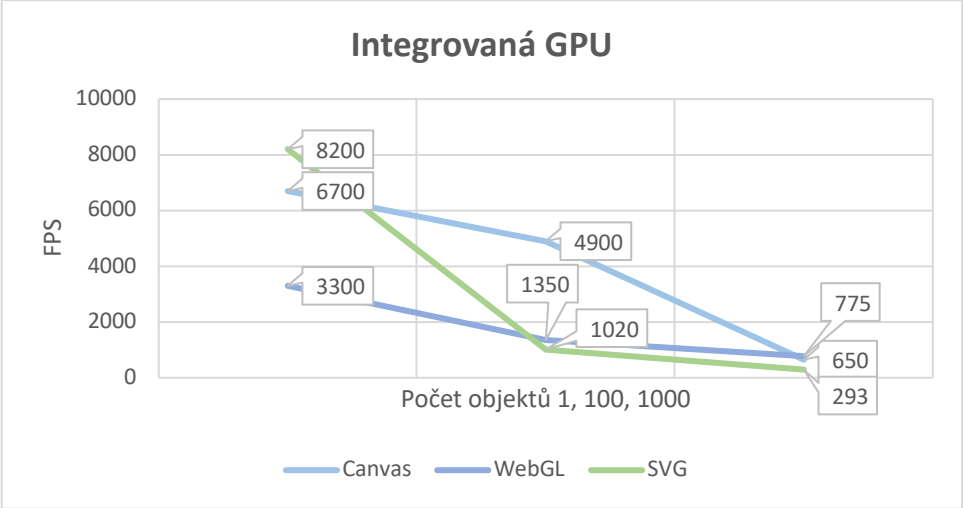
Z pohledu návrhu uživatelského rozhraní lze na internetu nalézt velké množství odborných článků zdůrazňujících význam responzivního a mobilně orientovaného designu. Přesto však mnoho vývojářů nemá dostatečné povědomí o osvědčených postupech v oblasti UI a UX. Problém je umocněn velkým množstvím různých zařízení, operačních systémů a webových prohlížečů, které je nutné zohlednit, včetně méně rozšířených alternativ k nejpoužívanějším prohlížečům, jako jsou Google Chrome nebo Microsoft Edge. Další komplikaci představují časté aktualizace, kdy může vlivem změn v prohlížeči nebo standardech přestat fungovat dříve funkční část kódu. Vzhledem k tomu, že chytré telefony dnes používá více než pět miliard uživatelů po celém světě, je nezbytné kombinovat zásady kvalitního programování s technologiemi, které jsou navrženy s ohledem na dlouhodobou udržitelnost a kompatibilitu s budoucími aktualizacemi (Shapiro, 2015).

Použití CSS animací je obecně považováno za optimální řešení pro jednoduché animační efekty, zejména z důvodu rychlé odezvy a minimální režie. Rychlá odezva je klíčovým faktorem pozitivního uživatelského zážitku, přičemž je přímo závislá na efektivní optimalizaci výkonu, rozumném využití operační paměti a grafického procesoru. Animace musí být navrženy s ohledem na omezené hardwarové prostředky mobilních zařízení, což klade zvýšené nároky na vývojáře. Vhodně zvolené vizuální efekty mohou zároveň výrazně přispět ke zlepšení celkového dojmu z aplikace, pokud jsou použity uvážlivě a s ohledem na výkon (Lersen, 2018).

Důležitou otázkou při návrhu animací je rovněž volba vhodné technologie, konkrétně zda použít SVG, Canvas nebo WebGL. Cílem není určit, která z uvedených technologií je obecně nejlepší, ale která je nejvhodnější pro konkrétní scénář použití. Canvas je HTML prvek určený pro kreslení grafiky v rastrové podobě a je často využíván pro grafy, koláže nebo jednodušší animace. SVG funguje jako kontejner pro vektorovou grafiku, kde se pracuje s přesně definovanými souřadnicemi a zobrazovací oblastí jednotlivých prvků. WebGL je technologie umožňující využití grafických shaderů přímo v prohlížeči a je určena pro velmi komplexní a výpočetně náročné grafické scénáře (Habr, 2025).

Při srovnávacím testování výkonu zmíněných technologií byly analyzovány scénáře s jedním, sto a tisícem objektů. Výsledky ukázaly, že při práci s jedním objektem je nejefektivnějším řešením

SVG, které vykazuje nejnižší zátěž procesoru i grafického jádra. Při zvýšení počtu objektů na sto zůstává nejefektivnější technologií Canvas, který nabízí dobrý výkon a zároveň rozsáhlý ekosystém knihoven a hotových řešení. Při práci s tisícem a více objekty se jako nejvhodnější řešení ukazuje WebGL, zejména v aplikacích podobných grafickým editorům, jako je například Figma (Habr, 2025).



Obr. 8: Rychlost vykreslování objektů (Integrovaná GPU)

Zdroj: vlastní zpracování dle Habr (2025)

Z uvedených výsledků vyplývá, že v projektech s omezeným počtem objektů, zpravidla do sta, je nejvhodnější volbou SVG, které minimalizuje zátěž CPU i GPU. Pro středně náročné scénáře s vyšším počtem objektů je efektivnější využití Canvasu, zatímco pro vysoce komplexní aplikace s velkým množstvím grafických prvků je nejlepším řešením WebGL. Správná volba technologie tak představuje klíčový faktor pro dosažení optimálního výkonu zejména na mobilních zařízeních (Habr, 2025).

V našich projektech je volba technologie SVG optimální, protože neumožňujeme uživatelům definovat libovolný počet objektů. Animace, které vytváříme, nejsou jednoduché, což znamená, že systém vyžaduje složité matematické výpočty pro přesné řízení pohybu jednotlivých prvků. Dalším důvodem je plánované využití obsluhy událostí, například kliknutí na konkrétní SVG elementy, což není prakticky možné realizovat v případě Canvasu, kde není přímá podpora událostí pro jednotlivé objekty. SVG navíc poskytuje nezávislost na rozlišení a umožňuje import z programů, jako je Affinity, s kompletními souřadnicemi a zachovanou strukturou skupin, což je mnohem efektivnější než ruční sestavování scény pomocí Canvas API (Habr, 2025).

Při volbě SVG oproti 3D grafice nebo videu je nejdůležitějším faktorem velikost a efektivita. SVG je textový XML kód, jehož velikost zpravidla nepřekračuje 100 kB, zatímco video ve vysokém rozlišení může dosahovat velikosti 10 až 50 MB, protože je tvořeno sekvencí rastrových snímků. V případě 3D obsahu je nutné načítat textury, modely a shadery, což často vede k velikosti souborů přesahující stovky megabajtů (Habr, 2025).

Z hlediska výpočetní náročnosti a energetické zátěže na mobilních zařízeních má video i 3D velký dopad. Video vyžaduje neustálé dekódování datového proudu, což zatěžuje baterii, a přitom je interaktivita velmi omezená. U 3D aplikací je nutné provádět překreslování scény až 60krát za sekundu, což výrazně zvyšuje energetickou náročnost zařízení. Naopak SVG animace

umožňuje převést statickou vektorovou grafiku do interaktivního webového rozhraní s minimální zátěží pro procesor a baterii (Habr, 2025).

JavaScriptová knihovna, jako je GSAP, poskytuje komplexní kontrolu nad složitými animacemi a jejich sekvencemi. Je vhodné používat specializované knihovny pro animace podél cest, morfing nebo pro sekvenční animaci více prvků. Volba knihovny se stává nezbytnou v projektech, kde se využívá časová osa, dynamické hodnoty nebo fyzikální simulace pohybu (OpenReplay, 2025).

Při práci s exportovaným SVG z programů, jako jsou Adobe Illustrator nebo Affinity Designer, je doporučeno využít nástroje pro optimalizaci SVG souborů. Nástroje umožňují odstranit nadbytečné desetinné hodnoty, sjednotit podobné příkazy cest, zjednodušit křivky a snížit tak velikost souboru. Pro opakující se tvary je vhodné používat element <use>, který výrazně zmenšuje velikost SVG souboru (OpenReplay, 2025).

Dále je doporučeno optimalizovat SVG soubory odstraněním metadat, komentářů a nadbytečných atributů. Vždy je také nutné respektovat preference uživatele a přidávat ARIA kontext pro podporu čteček obrazovky (OpenReplay, 2025).

```
gsap.fromTo("#path1",
  { drawSVG: "0%" },
  { drawSVG: "100%", duration: 2, ease: "power2.inOut" }
);

document.getElementById("button1").addEventListener("click", () => {
  gsap.to("#path1", { rotation: 360, duration: 1 });
});
```

Obr. 9: Animace SVG cesty s DrawSVGPluginem
Zdroj: vlastní zpracování dle OpenReplay (2025)

Pro zajištění kompatibility napříč prohlížeči je vhodné testovat aplikaci například pomocí nástrojů jako BrowserStack a vždy mít připravenou fallback strategii, například zobrazení statického obrázku místo interaktivní animace pro starší nebo nekompatibilní prohlížeče (OpenReplay, 2025).

2 Praktická část

Druhá, prakticky zaměřená část bakalářské práce plynule navazuje na teoretická východiska a detailně popisuje proces transformace historických podkladů do podoby funkčních interaktivních modelů. Text je strukturován chronologicky podle fází vývoje: od prvotní digitalizace ručních skic zadavatele, přes návrh architektury samotného webového prostředí, až po programové oživení modelů pomocí jazyka JavaScript a animační knihovny GSAP. Cílem dané kapitoly je poskytnout komplexní vhled do technického řešení, zdůvodnit zvolené implementační postupy a ukázat proces tvorby edukativní vrstvy projektu.

2.1 Příprava grafických podkladů

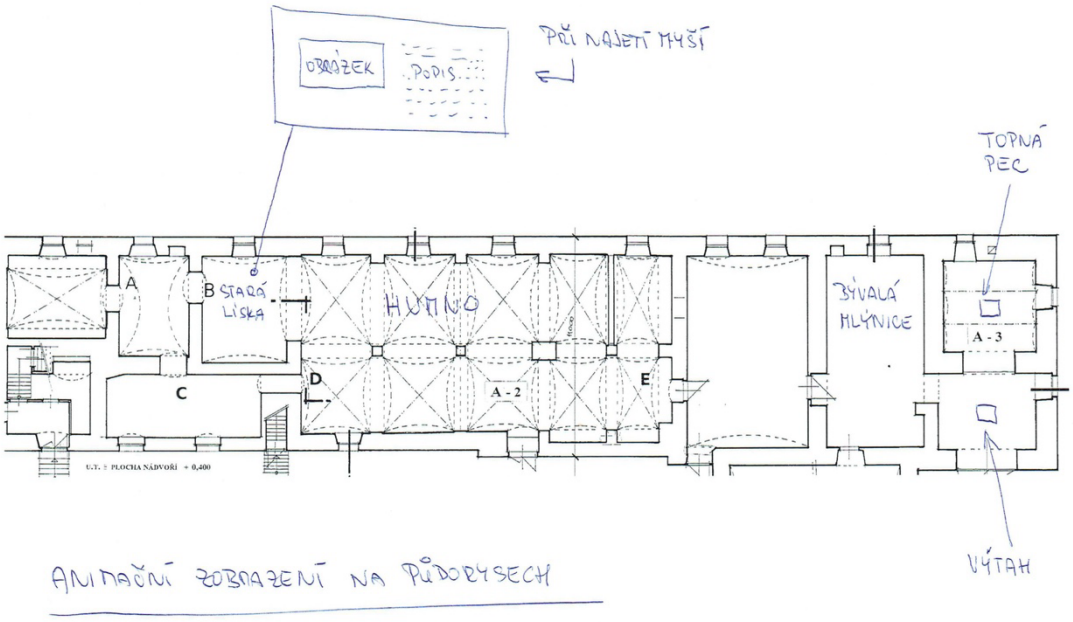
Klíčovým prvkem projektu je jeho grafická složka, která představuje hlavní nosný pilíř pro pochopení problematiky a vyžadovala důkladný metodický přístup. Vzhledem k tomu, že modely mají pro cílovou skupinu edukativní charakter, bylo nezbytné dbát na jejich historickou autenticitu. Toho bylo dosaženo aplikací dobových barevných palet v kombinaci s moderními technologiemi, což umožňuje atraktivní a srozumitelnou prezentaci historických procesů výroby sladu.

2.1.1 Realizace grafických podkladů a technická analýza

Východiskem pro praktickou část bylo sedm ručně kreslených technických skic poskytnutých zadavatelem. Podklady obsahovaly historicky věrný půdorys objektu s vyznačením klíčových technologických prvků, jako jsou výtah, prostory pro sušení sladu, pec a zařízení pro předčištění zrna. Součástí zadání byla rovněž rámcová definice animací a interaktivních prvků, určující směr pohybu objektů a jejich responzivitu na uživatelské podněty (click events). (MDN, 2026)

Navzdory poskytnutým instrukcím vyžadovala vizualizace hlubší pochopení technologických procesů výroby sladu, bez něhož by interpretace skic nebyla proveditelná. Z tohoto důvodu byla provedena rešerše historických pramenů zaměřená na tradiční postupy sladování, přičemž hlavní oporu tvořily materiály reflektující historickou praxi v českém a anglickém kontextu.

Hlavním cílem bylo na základě získaných znalostí transformovat skici do vektorového formátu SVG, optimalizovaného pro následnou implementaci do zdrojového kódu. Díky syntéze historických dat a technických nákresů bylo možné vytvořit vizuálně přesný a funkční model, který splňuje nároky na technickou i historickou validitu.

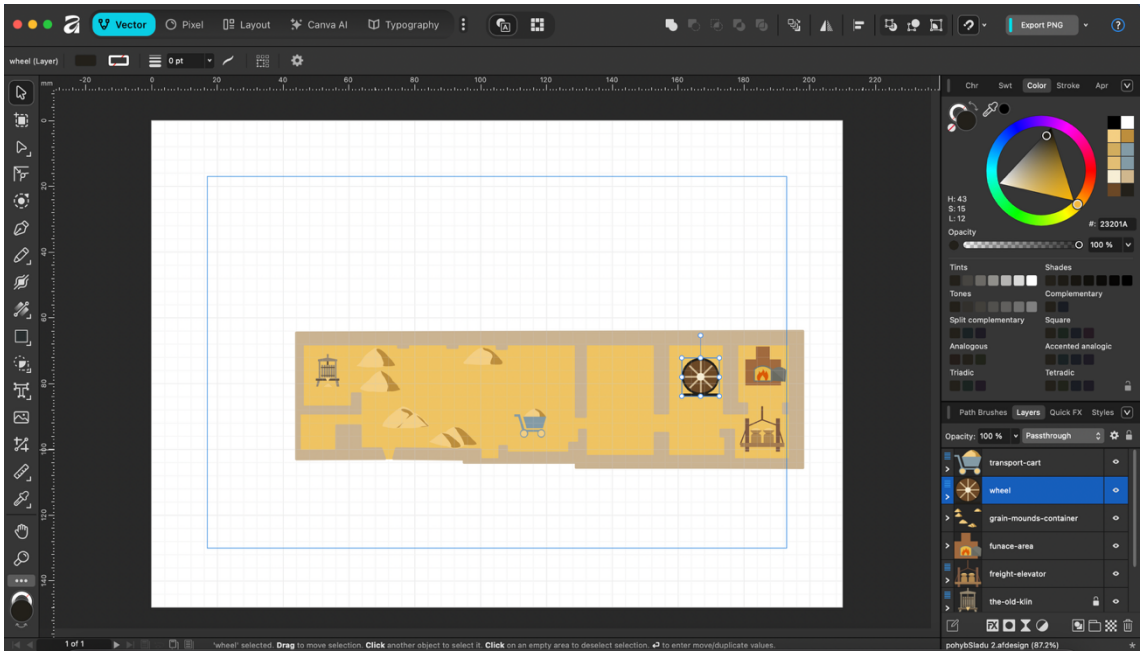


Obr. 10: Skic poskytnutý zadavatelem
Zdroj: zadavatel

2.1.2 Volba softwarového nástroje

Pro tvorbu grafických podkladů byl zvolen program Affinity Designer, který představuje plnohodnotnou alternativu k průmyslovému standardu Adobe Illustrator. Oba nástroje jsou určeny pro profesionální práci s vektorovou grafikou a technickou ilustraci, což bylo pro potřeby daného projektu klíčové. Výběr Affinity Designeru byl podmíněn několika faktory:

1. Software nabízí komplexní sadu nástrojů srovnatelnou s konkurenčními produkty, přičemž v současné době představuje dostupnější řešení pro akademické účely a běžné uživatele, aniž by docházelo k omezení kvality výstupu v profesionálním prostředí.
2. Na rozdíl od rastrových editorů (např. Adobe Photoshop) pracuje Affinity Designer primárně s vektorovou grafikou. Přístup byl zvolen z důvodu nutnosti zajistit responzivitu projektu. Výsledné ilustrace musí být ostře vykresleny při jakémkoli měřítku na různých zařízeních, od mobilních telefonů přes tablety až po desktopové monitory.
3. Zásadním kritériem byla schopnost softwaru exportovat čistý a validní SVG kód. Formát umožňuje přímou implementaci grafiky do zdrojového kódu aplikace, což je nezbytným předpokladem pro následnou tvorbu animací a programování interaktivních prvků, jako jsou modální okna
reagující na uživatelské podněty.
4. Pokročilá práce s vrstvami v rámci programu umožňuje efektivní seskupování objektů. Struktura je v pozdějších fázích projektu využívána k adresování konkrétních elementů pomocí identifikátorů nebo tříd, což usnadňuje manipulaci s grafikou skrze skripty.



Obr. 11: Prostředí Affinity na počítače

Zdroj: vlastní

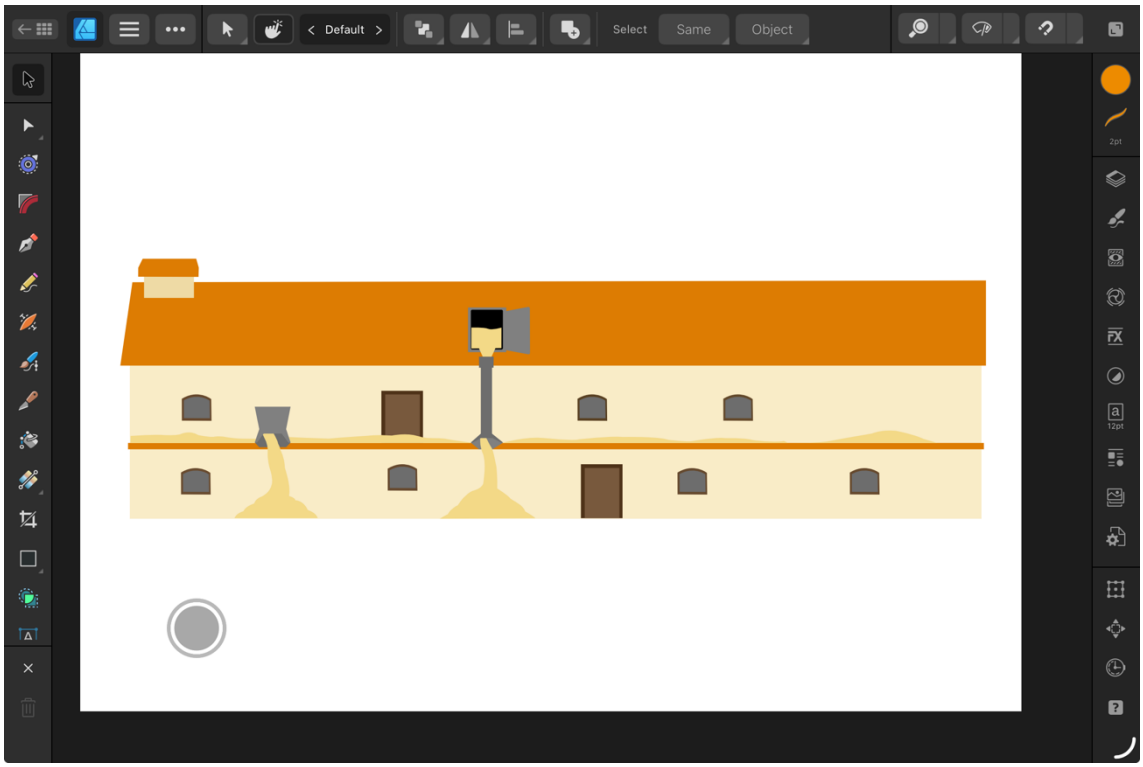
2.1.3 Proces vektorizace a digitální rekonstrukce

Vlastní tvorba modelů vycházela z technických skic poskytnutých zadavatelem, což výrazně zvýšilo efektivitu celého procesu digitalizace. Importované podklady sloužily v prostředí Affinity Designeru jako referenční vrstvy pro přímé překreslování, zejména v případě hlavní architektury budov. Postup byl zvolen za účelem zachování historické autenticity a přesných proporcí objektu.

Estetické a funkční pojetí: Pro grafické zpracování byl zvolen styl technické ilustrace, který se vyznačuje následujícími parametry:

- Čistota přímky a střídmá barevná paleta: Cílem bylo eliminovat vizuální šum a soustředit pozornost uživatele na samotný technologický proces.
- Absence hlubokých stínů a textur: Minimalistické pojetí napomáhá srozumitelnosti složitých struktur výroby sladu.
- Abstrakce a simplifikace: V rámci procesu byla uplatněna metoda řízeného zjednodušení. Byly odstraněny detaily postrádající sémantický význam (např. dobové poškození zdiva, nečistoty při zpracování zrna), které by mohly odvádět pozornost od edukačního obsahu.

Přístup umožnil vytvořit vizuálně koherentní systém, kde každý zobrazený prvek nese konkrétní informaci o výrobním postupu.

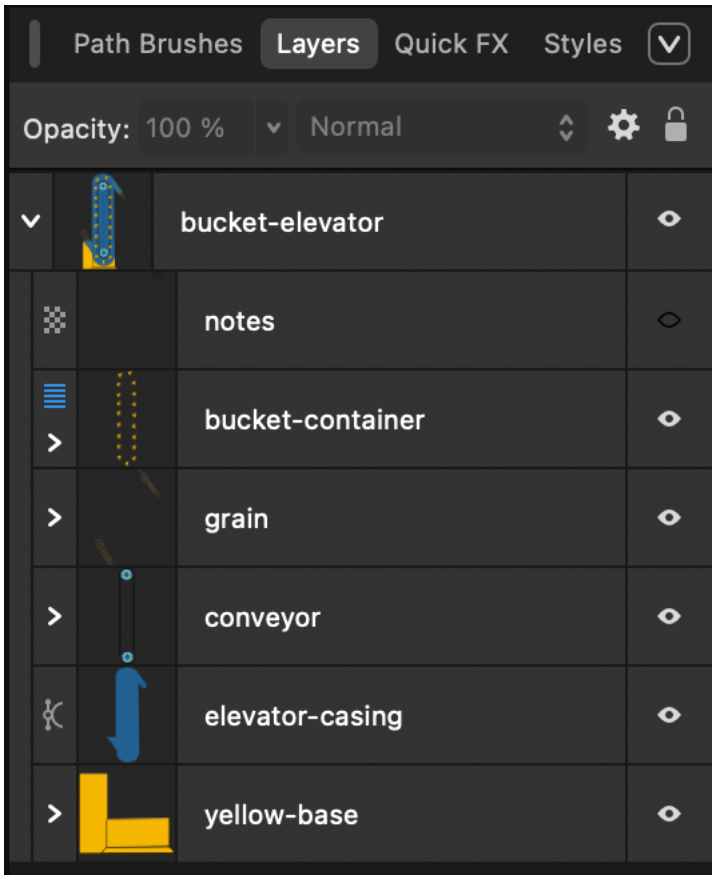


Obr. 12: Prostředí Affinity Designer 2 pro iPad
Zdroj: vlastní

2.1.4 Příprava hierarchie vrstev pro animaci

Klíčovým předpokladem pro následnou fázi oživení modelu byla systematická organizace vnitřní struktury dokumentu. Pro dosažení plynulé animace a možnosti programového ovládání jednotlivých částí bylo nutné již v grafickém editoru definovat logickou hierarchii a správnou nomenklaturu všech objektů. Bez dané přípravy by výsledný SVG kód tvořil monolitický celek, který by neumožňoval adresování konkrétních prvků a výrazně by komplikoval jakékoli budoucí revize.

V rámci struktury byly funkčně související elementy, například jednotlivé kbelíky korečkového výtahu, sdruženy do logických skupin. Každá vrstva a skupina obdržela unikátní sémantický název, který v exportovaném kódu slouží jako identifikátor nebo třída. Namísto automaticky generovaných názvů byly použity popisné termíny, což umožňuje práce v prostředí JavaScriptu či CSS přesně zacílit na konkrétní součást modelu a definovat její interaktivní chování. Důsledná příprava vrstev představuje nezbytný most mezi statickou ilustrací a funkční interaktivní aplikací.



Obr. 13: Logické skupiny vrstev
Zdroj: vlastní

2.1.5 Export a optimalizace dat

Závěrečná fáze přípravy grafických podkladů spočívala v technickém exportu do formátu SVG, který vyžadoval striktní dodržení několika parametrů pro zajištění maximální kvality a výkonu výsledné aplikace. Před samotným exportem bylo nezbytné validovat, zda jsou veškeré rozměry ilustrace definovány v celých pixelech. Krok je klíčový pro eliminaci nežádoucího rozostření, antialiasingu, hran při následném škálování v prohlížeči. Pro zajištění vysoké vizuální ostrosti na displejích s vysokým rozlišením byla nastavena hodnota DPI na minimum 144. Veškeré objekty byly zároveň převedeny na křivky, čímž bylo zamezeno případným interpretačním chybám nebo nechtěným změnám geometrie při vykreslování v různých grafických enginech.

Proces konfigurace exportního modulu v programu Affinity Designer byl podřízen potřebám následné editace zdrojového kódu. Byla aktivována funkce pro vkládání konců řádků, což zajišťuje logickou strukturu a sémantickou čitelnost kódu, která je nezbytná pro manuální úpravy a programování animací. (GSAP docs, 2026)

V rámci optimalizace velikosti souboru byly z exportu systematicky vyloučeny veškeré metadatové záznamy a skryté pomocné vrstvy. Uvedeným postupem došlo k výrazné redukci datového objemu souboru bez ztráty vizuální kvality. Výsledkem je čistý a validní kód, který je plně optimalizován pro plynulou integraci do vývojového prostředí a pro dosažení vysokého výkonu při vykreslování na straně klienta.

source-files > svgs > 02-bucketElevator.svg

You, 2 weeks ago | 1 author (You)

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/
3  <svg width="100%" height="100%" viewBox="0 0 2172 3071" version="1.1" xmlns
4      <g id="bucket-elevator">
5          <g id="yellow-base">
6              <rect id="hopper-right-part" x="351.028" y="2153.055" width="31
7              <path id="hopper-upper-part" d="M1364.091,2656.283c-1.911,0.497
8              <path id="hopper-down" d="M1367.334,2977.65l57.354,73.975c0,0 -
9              <path id="inlet-chute-outline-curve" d="M348.899,2150.513l3.508
10         </g>
11         <path id="elevator-casing" d="M1338.329,531.549c-0.082,9.886 -0.159
12         <g id="conveyor">
13             <g id="pulley-bottom">
14                 <path id="outer-pulley-bottom" d="M862.645,2533.861c0,0 6.7
15                 <path id="pulley-bottom-axis" d="M868.174,2542.685l295.626,
16                 <ellipse id="inner-pulley-bottom" cx="1011.742" cy="2536.68
17                 <ellipse id="middle-pulley-bottom" cx="1010.215" cy="2534.2
18             </g>
```

Obr. 14: Logické skupiny vrstev v kódu

Zdroj: vlastní

2.2 Technická implementace a architektura

Kapitola se věnuje podrobnému popisu programové struktury projektu a zdůvodnění volby konkrétních technologických nástrojů. Hlavním kritériem při návrhu architektury byla zajištění vysoké míry interaktivity, plynulosti animací a celkové výkonové optimalizace aplikace. Z uvedeného důvodu byl zvolen přístup založený na kombinaci standardů HTML5, CSS3 a nativního JavaScriptu (Vanilla JS), doplněný o specializovanou knihovnu GreenSock Animation Platform (GSAP).

Standardy HTML5 a CSS3 tvoří základní sémantickou a vizuální vrstvu projektu. Využití moderních vlastností CSS3 umožňuje efektivní definování stylů a responzivního rozvržení, zatímco HTML5 zajišťuje validní strukturu dokumentu nezbytnou pro správnou interpretaci SVG elementů.

Nativní JavaScript byl vybrán jako řídicí logika aplikace pro svou rychlost a absenci zbytečné režie, kterou by do projektu vnesly komplexní frameworky. JavaScript zajišťuje manipulaci s DOM strukturou, obsluhu uživatelských událostí a logické provázání jednotlivých částí modelu. (MDN, 2026)

Pro realizaci pokročilých sekvenčních animací byla do projektu integrována knihovna GSAP. Volba byla podmíněna především její vysokou efektivitou při vykreslování složitých vektorových scén a schopností precizně synchronizovat pohyby jednotlivých částí modelu. GSAP nabízí robustní nástroje pro časování a kontrolu animací, které standardní CSS animace v takovém rozsahu neumožňují, přičemž si zachovává minimální dopad na výkon procesoru koncového zařízení.

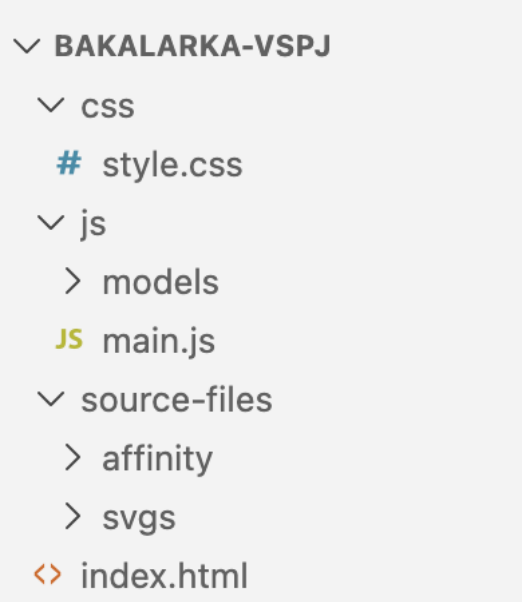
2.2.1 Struktura projektu

Pro zajištění přehlednosti a budoucí udržitelnosti kódu byla zvolena standardní modulární architektura webového projektu.

Díky využití CDN pro načítání externích knihoven (GSAP, ScrollTrigger) nebylo nutné fyzicky ukládat soubory frameworku do lokálního úložiště. To výrazně redukuje velikost celého projektu a zjednodušuje jeho distribuci.

Výsledná adresářová struktura vypadá následovně:

- Kořenový adresář: Obsahuje pouze soubor index.html, který definuje sémantickou kostru stránky a slouží jako kontejner pro SVG modely.
- Složka css: Obsahuje soubor style.css, který definuje globální styly, typografii a rozvržení stránky.
- Složka js: Zde je umístěn soubor main.js. Soubor obsahuje veškerou logiku animací, nastavení časových os a posluchače událostí. Externí knihovny nejsou přítomny, jsou volány dynamicky v HTML.



Obr. 15: Skupiny složek projektu
Zdroj: vlastní

2.2.2 Integrace SVG do HTML

Pro vložení vizuálních modelů do struktury webové stránky byla zvolena metoda přímého vnoření, inline SVG, namísto běžného využití značky <img>. Přístup byl zvolen z fundamentálního důvodu: značka <img> interpretuje vektorový soubor jako uzavřený externí objekt, u něhož není možné přistupovat k jeho vnitřní struktuře. V takovém případě by grafika

plnila pouze roli statické ilustrace bez možnosti jakékoli dynamické interakce či programového řízení.

Metoda inline SVG naopak integruje kompletní kód vektorového obrazu přímo do dokumentu HTML5, čímž se veškeré grafické elementy stávají součástí DOM stromu. To umožňuje vývojáři skrze skriptovací jazyk JavaScript či animační knihovnu GSAP adresovat konkrétní geometrické tvary, cesty a skupiny. (MDN, 2026)

V praktické rovině implementace dovoluje oživit specifické technologické detaily, které by jinak zůstaly statické. Příkladem jsou mechanické převody uvnitř obilného výtahu, konkrétně pohyb ozubeného kola, animace vozíku přepravujícího sladinový materiál nebo vizualizace dynamického ohně v interiéru pece. Díky přímému přístupu k identifikátorům těchto prvků bylo možné definovat jejich pohybové trajektorie, rychlost a reakce na akce uživatele, čímž bylo dosaženo komplexního edukačního zážitku.

Obr. 16: Ukázka Id u objektů

Zdroj: vlastní

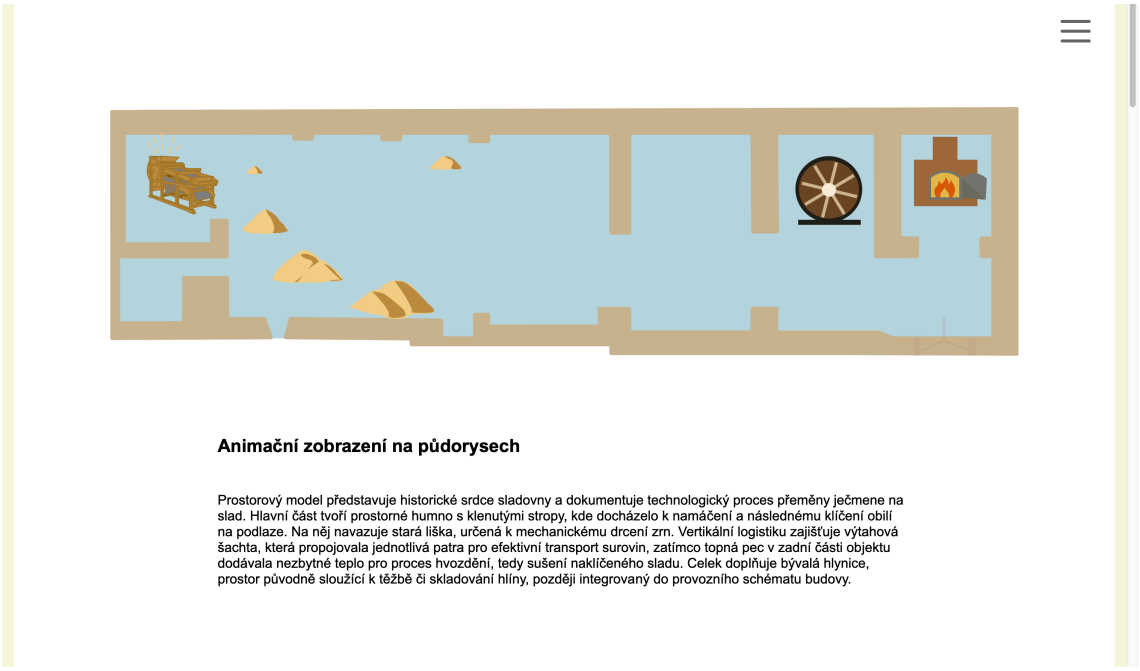
2.2.3 Stylování a layout

Pro precizní umístění animovaných modelů v rámci uživatelského rozhraní byl využit modul CSS Flexible Box Layout. Přístup umožnil vytvořit dynamický kontejner, který efektivně spravuje distribuci prostoru a zarovnání prvků. Konkrétně byly aplikovány vlastnosti flex-direction: row pro horizontální řazení prvků, doplněné o align-items: center a justify-content: center. Kombinace zajišťuje striktní centrování grafického obsahu ve všech osách, což je zásadní pro vizuální stabilitu interaktivní schématu.

Ačkoliv byly v raných fázích vývoje zvažovány externí knihovny, finální řešení se jich z důvodu minimalizace režie kódu a maximální kontroly nad stylováním zřeklo.

Díky vektorové podstatě grafiky a definování rozměrů pomocí pravidel width: 100 % a height: 100 vh se schémata automaticky přizpůsobují aktuální šířce zobrazovacího zařízení. Mechanismus zaručuje, že technologické schéma výroby sladu zůstává čitelné a funkční na

širokém spektru zařízení, od mobilních terminálů až po monitory s vysokým rozlišením, aniž by docházelo k deformaci obrazu nebo ztrátě kvality detailů.



Obr. 17: Počítačová verze stránky

Zdroj: vlastní

Obr. 18: Mobilní verze stránky

Zdroj: vlastní

2.2.4 Zapojení knihoven

Pro zajištění plynulého chodu animací byla do projektu integrována knihovna GSAP. Využití externí knihovny namísto standardních CSS animací bylo zvoleno z důvodu potřeby komplexního časování a lepší kontroly nad výkonem. (GSAP docs, 2026)

Knihovna je připojena prostřednictvím sítě CDN, což zajišťuje rychlejší načítání díky cachování v prohlížeči uživatele. (GSAP docs, 2026)

Kromě jádra gsap.min.js byl připojen také zásuvný modul ScrollTrigger.min.js. Plugin je klíčový pro optimalizaci uživatelského zážitku, zajišťuje, aby se náročné animace spustily až v okamžiku, kdy se daný model objeví ve viditelné části obrazovky, čímž se šetří výpočetní výkon zařízení. (GSAP docs, 2026)

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/gsap/3.12.5/gsap.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/gsap/3.12.5/MotionPathPlugin.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/gsap/3.12.5/ScrollTrigger.min.js"></script>
<script type="module" src="js/main.js"></script>
```

Obr. 19: Zapojení knihoven

Zdroj: vlastní

2.3 Oživení modelů a implementace animací

Po dokončení přípravy grafických podkladů a definování struktury dokumentu následovala fáze programování dynamického chování modelu. Pro účel práce byla integrována knihovna GSAP, která představuje standard v oblasti profesionálních webových animací. Hlavním důvodem pro volbu tohoto nástroje byla jeho schopnost vytvářet komplexní, časově synchronizované sekvence, které jsou spravovány pomocí objektů typu Timeline. (GSAP docs, 2026)

Využití mechanismu GSAP Timeline umožnilo precizní orchestraci jednotlivých animačních kroků v rámci technologického procesu výroby sladu. Namísto izolovaných animací, které by bylo obtížné vzájemně koordinovat, umožňuje Timeline řetězit pohyby jednotlivých elementů, například synchronizaci pohybu obilného výtahu s navazujícím procesem čištění zrna. Přístup zajišťuje, že se celá vizualizace chová jako jeden plynulý celek, kde na sebe jednotlivé fáze logicky a časově navazují. (GSAP docs, 2026)

Knihovna GSAP byla rovněž zvolena pro svou vysokou výkonovou optimalizaci při manipulaci s vlastnostmi SVG objektů. Díky pokročilé kontrole nad časováním, easing, možností pozastavení, opětovného spuštění nebo reverzního chodu sekvencí, bylo možné dosáhnout vysoké míry interaktivity a didaktické srozumitelnosti celého modelu. Programová logika tvoří jádro praktické části, které transformuje statickou technickou ilustraci v dynamický edukační nástroj. (GSAP docs, 2026)

2.3.1 Selektování elementů v DOM

Základním předpokladem pro jakoukoli animaci je schopnost skriptu jednoznačně identifikovat grafické prvky, se kterými má manipulovat. Jelikož je formát SVG vložen přímo do HTML kódu, tzv. inline SVG, stávají se všechny jeho části od křivek po skupiny plnohodnotnými uzly v DOM. (MDN, 2026)

Pro výběr těchto elementů byly v jazyce JavaScript využity standardní metody document.querySelector() a document.querySelectorAll(). Při adresaci byl kladen důraz na logické rozlišení mezi unikátními a opakujícími se prvky. (MDN, 2026)

Pro specifické stroje nebo jejich hlavní části byly v grafickém editoru nastaveny unikátní id atributy. V kódu je pak k těmto prvkům přistupováno jako k jednotlivým objektům.

Pro elementy, které se ve scéně vyskytují v mnoha kopiích byly využity třídy.

Přístup je klíčový pro optimalizaci kódu. Místo toho, aby bylo nutné psát animaci pro každé z 30 zrnek zvlášť, stačí v knihovně GSAP zacílit na jednu společnou třídu. JavaScript pak automaticky aplikuje pohyb na všechny elementy s touto třídou současně, což výrazně redukuje objem psaného kódu a zvyšuje přehlednost.

```
export function initFloorPlan() {
    const GrainForCleaner = document.querySelectorAll(".grain-for-cleaner");
    const piles = document.querySelectorAll("#grain-mounds-container > g");
    const wheel = document.querySelectorAll(".wheel-part");
    const flames = document.querySelector("#flames");
    const elevator = document.querySelectorAll("#freight-elevator1 > g");
```

Obr. 20: Ukázka querySelector

Zdroj: vlastní

Použití selektorů uvedeným způsobem umožňuje knihovně GSAP efektivně distribuovat výkon a vytvářet komplexní choreografie pohybu bez zbytečného zatěžování prohlížeče.

2.3.2 Tvorba časové osy

Ústředním prvkem programové logiky je koncept GSAP Timeline, který lze analogicky přirovnat k montážní stopě v profesionálních video editorech, avšak implementovaný přímo v programovém kódu. Technologie umožňuje nelineární správu času a poskytuje absolutní kontrolu nad chronologií celého vizualizovaného procesu. (GSAP docs, 2026)

Využití časové osy v rámci projektu přináší několik klíčových technických výhod. Sekvenční a paralelní řízení umožňuje přesně definovat pořadí jednotlivých akcí. Lze určit, který objekt zahájí pohyb jako první a který na něj naváže s přesně definovanou prodlevou či v těsném závěsu. Stejně tak je možné konfigurovat paralelní procesy, kdy se současně pohybuje více elementů nezávisle na sobě, což je nezbytné pro simulaci plynulého provozu sladařské linky. (GSAP docs, 2026)

Modulace rychlosti a dynamiky elementu lze přiřadit specifické parametry rychlosti a časových funkcí, čímž je dosaženo realistického znázornění fyzikálních procesů, jako je sypání zrna nebo rotace mechanických částí. (GSAP docs, 2026)

Na rozdíl od standardních CSS animací umožňuje snadnou synchronizaci časové osy komplexních řetězců událostí. Pokud je potřeba upravit délku trvání jedné fáze, systém automaticky přepočítá a posune navazující akce, čímž je zachována integrita celého technologického toku. (GSAP docs, 2026)

Metodický přístup transformuje izolované pohyby v koherentní celek, kde je každý krok logicky provázán s předchozím, což významně přispívá k přehlednosti a edukační hodnotě celého modelu.

Pro synchronizaci složitých technologických procesů, jakými jsou sladovnické postupy, nebylo možné spoléhat na jednoduché CSS přechody ani na základní JavaScriptové časovače. Metody by při řazení více animací za sebou vyžadovaly složité matematické výpočty zpoždění a vedly by k nepřehlednému kódu.

Proto byla využita klíčová funkce časová osa. Koncept lze přirovnat k nelineárnímu střihu videa. Stejně jako střihač klade jednotlivé záběry na časovou osu za sebe, umožňuje GSAP programově definovat sekvenci událostí. (GSAP docs, 2026)

Timeline garantuje přesné načasování. Pokud se změní délka první animace, všechny následující se automaticky posunou, aniž by bylo nutné přepisovat zbytek kódu.

V projektu je instance časové osy inicializována ve stavu pozastaveno. Jinými slovy prohlížeč načte a připraví všechny animace do paměti, ale nespustí je okamžitě. Přehrávání je zahájeno až na základě interakce uživatele, např. scrollování k elementu. (GSAP docs, 2026)

```
const tl = gsap.timeline({ paused: true, repeat: -1, ease: "none"});      You, 5 days ago • feat: add timeline and pop up …

const GrainForCleaner = document.querySelectorAll(".grain-for-cleaner1");

gsap.set(GrainForCleaner, { strokeDasharray: "12.112, 9.69" });

tl.to(GrainForCleaner  , {
    strokeDashoffset: -21.802,
    duration: 1,
    repeat: -1,
    ease: "none"}, 0
);
const wheel = document.querySelectorAll(".wheel-part");

tl.to(wheel,
    {rotation: 360, ease: "none",duration: 10, repeat: -1, transformOrigin: "center"}, 0
);

tl.to(flames,
    {scaleY: 1.1, scaleX: 0.8, transformOrigin: "50% 100%", duration: 0.4, repeat: -1, yoyo: "true", ease: "rough"}, 0
);
```

Obr. 21: Ukázka časové osy

Zdroj: vlastní

2.3.3 Implementace klíčových mechanismů

V rámci vizualizace technologických procesů představuje jeden z klíčových mechanismů animace rotujících částí, například ozubených kol či jiných kruhových prvků mechanizace. Při realizaci daného pohybu v prostředí knihovny GSAP je nezbytné nejprve definovat referenční skupinu elementů, které tvoří daný objekt, a následně určit bod, kolem něhož se bude transformace odehrávat. K tomuto účelu slouží vlastnost transformOrigin, jejíž správné nastavení do těžiště objektu zajišťuje, že rotace o 360 stupňů probíhá symetricky a bez vizuálních deformací. (GSAP docs, 2026)

Z hlediska programové logiky a správy času v rámci komplexních sekvencí bylo nutné zvolit vhodný způsob iterace pohybu. V případech, kdy je animace součástí izolované časové osy určené pro trvalý chod, lze využít parametr repeat: -1, který zajišťuje nekonečnou smyčku. Pokud je však rotace integrována do hlavní časové osy spolu s dalšími navazujícími událostmi,

ukázalo se jako efektivnější využít metodu násobné rotace, například definováním hodnoty rotation: 360 * 8. (GSAP docs, 2026)

Přístup umožňuje exaktně vypočítat délku trvání animace v poměru k ostatním prvkům scény a zamezit tak nekonečnému zacyklení, které by mohlo zablokovat spuštění následujících animačních kroků. Uvedeným způsobem je v kódu zajištěna deterministická struktura, kde každý mechanický pohyb začíná a končí v přesně stanovený moment, což je zásadní pro plynulý a logický přechod mezi jednotlivými fázemi výroby sladu.

```
tl.to(wheel, {
    rotation: 360,
    ease: "none",
    duration: 10,
    repeat: -1,
    transformOrigin: "center"
}, 0);
```

Obr. 22: Ukázka kódu pro rotace kola

Zdroj: vlastní

Složitější technologický prvek představuje animace obilného výtahu ve třetím modelu, kde bylo nutné simulovat plynulý a nekonečný pohyb kbelíků s obilím směrem vzhůru. Technické řešení spočívá v iteraci přes pole grafických elementů v rámci dané skupiny, přičemž ke každému objektu je přistupováno individuálně na základě jeho indexu. Pro každý kbelík je definována interní časová osa s atributem nekonečného opakování, což umožňuje nezávislou správu pohybu každého jednotlivého kusu v rámci celého mechanismu. (GSAP docs, 2026)

V rámci inicializace jsou objekty nastaveny jako neviditelné, čímž je vytvořen základ pro iluzi jejich plynulého vstupu do scény, a jejich transformační body jsou vycentrovány v obou osách pro zajištění stability během pohybu. Celková délka trvání vnitřního animačního cyklu byla stanovena na 12 sekund. K definování pohybu byla využita metoda MotionPath, která pomocí vlastností path a align navádí elementy po přesně určené křivce a zároveň koriguje jejich rotaci tak, aby odpovídala směru stoupání výtahu. (GSAP docs, 2026)

Pro dosažení vizuální věrohodnosti jsou do vnitřní časové osy integrovány efekty prolnutí: v úvodních 0,8 sekundách dochází k postupnému zobrazení prvku, zatímco v závěru sekvence je aplikován inverzní efekt zmizení. Finální integrace do hlavní časové osy projektu je vyřešena pomocí dynamického výpočtu časového posunu (delay). Je definován jako podíl celkové doby trvání cyklu a celkového počtu kbelíků. Každý element tak obdrží specifické zpoždění odvozené od svého indexu, což ve výsledku vytváří dojem rovnoměrně rozprostřeného, kontinuálního toku materiálu v celém vertikálním dopravníku. (GSAP docs, 2026)

```
scoops.forEach((scoop, index) => {

    let scooptl = gsap.timeline({
        repeat: -1,
        defaults: { ease: "none" }
    });


    gsap.set(scoop, {
        opacity: 0,
        transformOrigin: "50% 50%"
    });

    const totalDuration = 12;


    scooptl.to(scoop, {
        duration: totalDuration,
        motionPath: {
            path: "#scoop-motionpath",
            align: "#scoop-motionpath",
            alignOrigin: [0.5, 0.5]
        }
    }, 0);


    scooptl.to(scoop, { opacity: 1, duration: 0.8 }, 0)
    scooptl.to(scoop, { opacity: 0, duration: 0.8 }, totalDuration - 0.8);

    tl.add(scooptl.delay(-index * (totalDuration / scoops.length)), 0);
    });
```

Obr. 23: Ukázka kódu pro motionPath

Zdroj: vlastní

K vizualizaci procesu padajícího zrna byla využita metoda animace přerušované čáry (stroke-dash). Technika je založena na vytvoření vektorové cesty s definovaným vzorem střídání plných částí a mezer, přičemž výsledného efektu pohybu je dosaženo plynulým posunem tohoto vzoru podél trajektorie. V rámci programové logiky dochází k cyklickému posunu o celou délku jednoho segmentu ve velmi krátkém časovém intervalu, což u pozorovatele vyvolává dojem kontinuálního toku materiálu směrem dolů. (GSAP docs, 2026)

Technické řešení spočívá v nastavení parametru strokeDashoffset na hodnotu odpovídající délce vzoru, která v tomto konkrétním případě činí -16,52. Volba záporné hodnoty je zde klíčová pro určení směru vektoru pohybu, tedy vertikálního pádu dolů. Celá sekvence je nastavena na krátkou dobu trvání 0,6 sekundy s nekonečným opakováním (repeat: -1). Minimalistický, ale efektivní přístup umožňuje simulovat plynulý technologický proces bez nutnosti animovat stovky jednotlivých objektů (zrn), což výrazně snižuje výpočetní náročnost na straně klientského prohlížeče a zajišťuje plynulost animace i na méně výkonných zařízeních. (GSAP docs, 2026)

```
tl.to(grainPaths, {
    strokeDashoffset: -16.52,
    duration: 0.6,
    ease: "none",
    repeat: -1
}, 0);
```

Obr. 24: Ukázka kódu pro cyklické ději

Zdroj: vlastní

2.3.4 Interaktivita

Pro zvýšení dynamiky a uživatelského komfortu při prohlížení edukačního obsahu byl do projektu implementován mechanismus řízení animací na základě polohy skrolování (ScrollTrigger). Hlavním cílem tohoto řešení je zajistit, aby uživatel sledoval animační sekvence od jejich počátku, a zároveň optimalizovat využití systémových prostředků koncového zařízení. (GSAP docs, 2026)

Technické řešení spočívá v přiřazení individuální časové osy a unikátního indexu každému grafickému modelu. V základním stavu jsou všechny animace pozastaveny. Mechanismus detekce viditelnosti neustále monitoruje pohyb uživatele na stránce a v okamžiku, kdy je v zorném poli (viewportu) identifikováno alespoň 20 % plochy nadcházejícího bloku, dojde k automatickému spuštění příslušné animační sekvence. Proces funguje i reverzně: pokud viditelná část předchozího bloku klesne pod hranici 20 %, animace je okamžitě pozastavena. Přístup výrazně snižuje zátěž procesoru, neboť eliminuje vykreslování náročných vektorových operací v neviditelných částech webové stránky. (GSAP docs, 2026)

Důležitým prvkem z hlediska kontinuity uživatelského zážitku je funkce zachování stavu animace. Pokud se uživatel v rámci navigace vrátí k předchozímu modelu, systém identifikuje místo přerušení a plynule naváže na animační proces v bodě, kde byl dříve pozastaven. Díky této logice nedochází k neustálému restartování sekvencí, což přispívá k plynulosti prezentace a umožňuje uživateli studovat historické technologické postupy vlastním tempem bez ztráty kontextu. (GSAP docs, 2026)

```
if (tl) {
    ScrollTrigger.create({
        trigger: page,
        start: "top 80%",
        end: "bottom 20%",      You, now • Uncc
        onEnter: () => {
            tl.play();
        },
        onLeave: () => tl.pause(),
        onEnterBack: () => tl.play(),
        onLeaveBack: () => tl.pause(),
        markers: false
    });
}
```

Obr. 25: Ukázka kódu pro ScrollTrigger

Zdroj: vlastní

Pro prohloubení edukačního potenciálu prvního modelu integrovány interaktivní zóny, které uživateli umožňují získat detailnější informace o specifických částech technologického zařízení. Indikace interaktivity je řešena na úrovni CSS změnou kurzoru na symbol ukazovátka (pointer) při najetí na příslušný objekt. Vizuální podnět slouží jako intuitivní nápověda, že daný prvek reaguje na uživatelský vstup.

Z hlediska designu uživatelského rozhraní bylo rozhodnuto využít modální vyskakovací okna namísto drobných plovoucích nápověd, tooltips. Rozhodnutí bylo podmíněno dynamickou povahou modelu; jelikož se řada objektů v rámci animace pohybuje, statická modální okna eliminují riziko ztráty kontaktu s kurzorem a zajišťují, že textové i obrazové informace zůstanou stabilně zobrazeny po celou dobu studia daného prvku. (MDN, 2026)

Technická realizace je postavena na využití posluchačů událostí (eventListener), které jsou navázány na konkrétní identifikátory v rámci DOM struktury SVG modelu. Po detekci kliknutí na vybraný element proběhne skript, který vyhledá odpovídající kontejnery uvnitř připravené šablony modálního okna. Do těchto kontejnerů je následně dynamicky vložen relevantní textový obsah a doprovodná grafika odpovídající vybranému stroji či části provozu. Modulární přístup umožňuje snadnou správu obsahu a zajišťuje plynulou interakci mezi grafickým modelem a informační vrstvou aplikace. (MDN, 2026)

```
const pileGroup = document.querySelector("#grain-mounds-container");

if (pileGroup) {
    pileGroup.style.cursor = "pointer";

    pileGroup.addEventListener ("click", () => {
    document.getElementById("popup-title").innerText = "Humno";
    document.getElementById("popup-text").innerText = "Prostorný sál s nízkou teplotou
    document.getElementById("popup-image-container").innerHTML = `
    <svg viewBox="470 340 180 100" width="100" height="100">
        <g id="pile5">
            <path d="M480.899,402.198c0,0 33.716,10.718 37.795,11.811c0,0 18.369,4.54
            <path d="M571.844,349.049c0,0 39.391,53.072 34.252,63.78c0,0 38.019,2.994
            <path d="M526.396,370.308c0,0 41.892,40.52 32.456,49.732c0,0 7.661,5.114
        </g>
    </svg>
    ` ;

    popup.classList.remove("popup-hidden");
});
```

Obr. 26: Ukázka pop-up

Zdroj: vlastní

3 Uživatelské testování

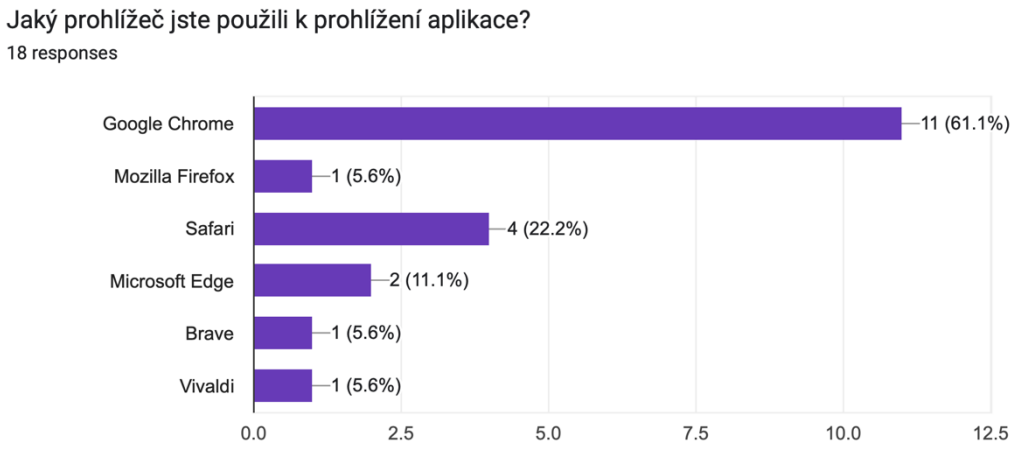
Kapitola prezentuje výsledky uživatelského testování vyvinuté webové aplikace, které bylo provedeno za účelem posouzení její funkčnosti a uživatelské přívětivosti. Hlavní pozornost je věnována analýze interakce uživatelů s animovanými prvky a identifikaci případných technických nedostatků. Na základě získaných dat jsou v závěru formulovány závěry o aktuálním stavu projektu a navrženy možnosti jeho dalšího rozvoje.

3.1 Cíle testování

Primárním cílem testování bylo zjistit, zda se podařilo model spustit na všech zařízeních a zda modely přispívají k lepšímu pochopení dění na nich samotných. Sekundárním cílem bylo posoudit plynulost animací a identifikovat oblasti pro další zlepšení uživatelského rozhraní.

3.2 Metodika testování

Testování probíhalo formou sběru odpovědí na deset otázek prostřednictvím služby Google Forms. Odkaz byl rozeslán osmnácti studentům vysokých škol v České republice, přičemž většina z nich studovala v Jihlavě. Jejich úkolem bylo otevřít odkaz na webovou stránku publikovanou na GitHubu, která obsahovala modely, a současně nebo následně vyplnit dotazník. U některých otázek bylo možné zvolit více než jednu odpověď. Základní charakteristiky skupiny lze rozdělit podle používaného prohlížeče a zařízení. Z hlediska zařízení dvanáct respondentů uvedlo, že si modely prohlíželo na telefonu, a sedm respondentů na počítači. Další klíčovou charakteristikou byl prohlížeč, jehož prostřednictvím respondenti stránku otevřeli. Nejpoužívanějším prohlížečem byl Google Chrome s jedenácti hlasy, druhým nejčastějším byl Safari se čtyřmi hlasy a třetím Microsoft Edge se dvěma hlasy. Zastoupeny byly i méně běžné prohlížeče, jako Brave, Vivaldi a Firefox, každý s jedním hlasem.



Obr. 27: Použité prohlížeči

Zdroj: vlastní

Každý testující odpověděl na následujících deset otázek:
1. Jaký prohlížeč jste použili k prohlížení aplikace?
2. Typ zařízení?
3. Jak srozumitelná je struktura stránky?
4. Jak logicky vnímáte posloupnost animací?
5. Byly zde prvky, jejichž účel vám nebyl jasný?

6. Jak plynulá se vám animace zdála?
7. Zaznamenali jste zpoždění, zasekávání nebo trhání?
8. Pomáhá animace lépe pochopit strukturu obsahu?
9. Celkové hodnocení aplikace:
10. Co by se dalo zlepšit?

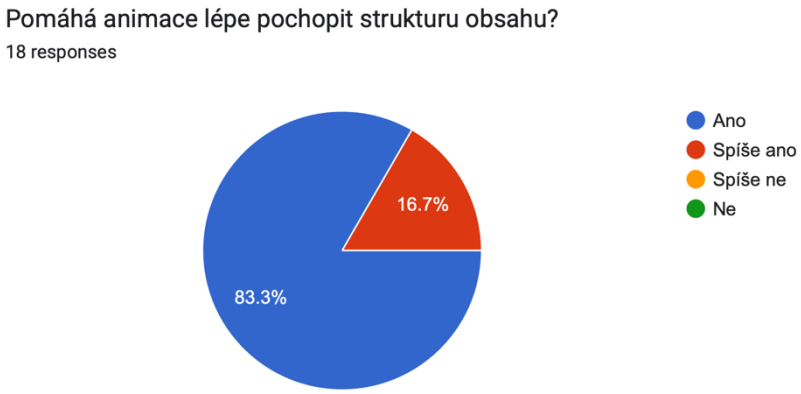
Otázky číslo jedna, dvě a pět umožňovaly výběr více možností. Respondenti mohli zvolit více odpovědí, pokud modely testovali na různých zařízeních nebo v několika prohlížečích. Otázky číslo tři, čtyři, šest a devět byly škálové, přičemž respondenti hodnotili míru souhlasu s negativním či pozitivním tvrzením na bodové škále, kde jednička představovala nejlepší, nejpozitivnější, hodnocení a pětka naopak hodnocení nejhorší. Skupina otázek byla zaměřena na to, jak dobře uživatel modelům rozumí, jak plynule fungují a na celkové hodnocení modelů i stránky. Otázky číslo pět a sedm byly dichotomické, ano nebo ne, a zjišťovaly, zda se respondent během prohlížení setkal s nějakými problémy. Poslední, desátá otázka, dávala respondentovi možnost otevřeně vyjádřit svůj názor na stránku jako celek a nabídnout náměty na zlepšení pro budoucí úpravy.

3.3 Výsledky testování

Na otázku týkající se srozumitelnosti struktury stránky patnáct respondentů uvedlo, že struktura je jasná, a dva respondenti odpověděli, že je v zásadě jasná. Na následující otázku, zda respondenti rozumí logické posloupnosti, čtrnáct uživatelů odpovědělo, že posloupnosti rozumí zcela, tři uvedli, že rozumí dostatečně, a jeden uživatel odpověděl, že rozumí průměrně. Na základě těchto odpovědí lze konstatovat, že struktura modelů i stránky je dostatečně srozumitelná, avšak existuje prostor pro dílčí zlepšení.

Následující otázky se týkaly technické realizace. Na otázku, zda byly animace plynulé, sedmnáct respondentů odpovědělo naprostým souhlasem a jeden respondent pouhým souhlasem, z čehož lze vyvodit, že všechny animace jsou dostatečně plynulé. Na otázku, zda bylo zaznamenáno trhání animací, sedmnáct respondentů odpovědělo ne, jeden uživatel odpověděl ano. Z toho vyplývá, že animace jsou téměř vždy plynulé a netrhají se, avšak ojediněle se problém může vyskytnout.

Předposlední skupina otázek byla zaměřena na vzdělávací potenciál a srozumitelnost modelů. Na otázku, zda se v modelech vyskytly nejasné prvky, sedmnáct uživatelů odpovědělo, že takové momenty nenastaly, a jeden uživatel uvedl, že se s nejasnými prvky setkal. Na otázku, zda animace pomáhají pochopit strukturu a dění v modelech, patnáct uživatelů jednoznačně odpovědělo ano, tři uživatelé odpověděli spíše ano, z čehož vyplývá, že animace a modely přispívají k lepšímu pochopení vnitřních procesů modelů a celkové problematiky výroby sladu.



Obr. 28: Struktura obsahu

Zdroj: vlastní

V poslední otevřené otázce, která se ptala na návrhy na zlepšení modelů, byly získány různé podněty v závislosti na osobních preferencích uživatelů. Například jeden z respondentů navrhl úpravy ve vizuální stránce modelů, konkrétně sjednocení stylu animací pomocí obrysů prvků, přidání tvarů, zlepšení okrajů a řešení plochého vzhledu obrázků. Další uživatelé se zaměřili na informační hodnotu a navrhli doplnit více textových popisů, které by usnadnily orientaci nově příchozím návštěvníkům stránky. V rámci zvýšení interaktivity se objevil podnět u prvního modelu, kde lze klikat na různé prvky, zavést barevné zvýrazňování oblastí a odpovídajících částí textu. Podle uživatele by pomohlo lépe pochopit, která oblast je v textu zrovna popisována, a usnadnilo by porozumění celému vysvětlení. Posledním významným návrhem bylo doplnění anglického překladu, což by umožnilo šířit informace mezi větší počet turistů a zahraničních návštěvníků.

Na základě dotazníkového šetření se celkové hodnocení a spokojenost s aplikací blíží maximálnímu možnému skóre. Téměř všechny prvky uživatelského rozhraní byly intuitivně srozumitelné, pouze jeden uživatel upozornil na existenci prvků, jejichž účel mu zůstal nejasný.

Závěr

Předložená bakalářská práce splnila stanovený cíl, kterým bylo vytvoření responzivní webové stránky s interaktivními modely historického technologického vybavení sladovny v Želetavě. Výsledné řešení slouží jako moderní nástroj pro propagaci kulturní památky Starý hrad a přispívá k uchování povědomí o mizejícím průmyslovém dědictví regionu.

V teoretické části byla provedena analýza sladovnických procesů, která se stala nezbytným podkladem pro logickou strukturu animací. Na základě komparace dostupných technologií byla pro realizaci zvolena kombinace vektorového formátu SVG a animační knihovny GSAP. Vybraná volba se v průběhu implementace ukázala jako optimální, zejména z hlediska vysokého výkonu, přesného časování komplexních sekvencí a minimální zátěže pro mobilní zařízení.

Praktický přínos práce spočívá v kompletním procesu digitální rekonstrukce, kdy byly původní ruční technické skici transformovány do podoby precizních vektorových modelů v programu Affinity Designer. Pomocí jazyka JavaScript a objektů GSAP Timeline byly modely oživeny tak, aby věrně simulovaly historické výrobní postupy, jako je pohyb korečkového výtahu.

Poslední fází projektu bylo uživatelské testování, které potvrdilo vysokou úroveň srozumitelnosti a technické stability aplikace. Respondenti vysoce hodnotili zejména plynulost animací a edukační potenciál modelů, které jim pomohly lépe pochopit vnitřní mechanismy sladovny. Testování zároveň definovalo směry pro budoucí rozvoj, mezi něž patří implementace anglické verze webu pro zahraniční návštěvníky nebo další zvýšení interaktivity propojením textového popisu s vizuálním zvýrazněním prvků.

Výsledná webová stránka představuje funkční celek, který propojuje technologie s regionální historií. Tudíž projekt může sloužit jako inspirace pro digitalizaci dalších technických památek.

Seznam použité literatury

Affinity Designer iPad – Official Learning Resources [online]. 2025 [cit. 2025-11-19]. Dostupné z: https://affinity.serif.com/en-gb/learn-v2/designer/ipad/

ANDERLE, Jan. Tradiční výroba sladu a piva z hlediska stavební typologie na příkladech z Plzeňska. In: *Památky západních Čech III - 2013* [online]. Plzeň: Jalna, 2013, s. 18-32 [cit. 2025-11-19]. Dostupné z: https://www.npu.cz/uop/plzen/pdf/publikacni%20cinnost/pzc-2013/pzc-iii-2013-sep.-02---vyroba-sladu-a-piva.pdf

BEŇO, Miroslav a Miroslav ÖLVECKÝ. Measuring the performance of techniques for dynamic 2D animation in web browsers. Journal of Applied Mathematics Statistics and Informatics [online]. 2024, 20(2), 77-110 [cit. 2025-11-19]. Dostupné z: doi:10.2478/jamsi-2024-0009

BORŮVKOVÁ, Jana, Stanislava DVOŘÁKOVÁ a Hana VOJÁČKOVÁ. *Jak psát práce na VŠPJ: Typografická pravidla pro studenty VŠPJ*. Jihlava: Vysoká škola polytechnická Jihlava, 2021. ISBN 978-80-88064-54-1

Citace.com [online]. [cit. 2025-11-23]. Dostupné z: citace.com

Cross Browser Compatibility Issues to Avoid. BrouserStack [online]. 2025 [cit. 2025-12-20]. Dostupné z: https://www.browserstack.com/guide/common-cross-browser-compatibility-issues

DOLEŽAL, Libor a Milan STAREC. Stavebně technologický průzkum vertikálního hvozdu ve vrchnostenské sladovně Želetava. *The Construction* [online]. 2015, 61(7-8), 221-235 [cit. 2025-11-19]. Dostupné z: doi:https://doi.org/10.18832/kp2015023

MDN [online]. 2026 [cit. 2026-03-12]. Dostupné z: https://developer.mozilla.org/en-US/

GSAP docs [online]. 2026 [cit. 2026-03-11]. Dostupné z: https://gsap.com/docs/v3/

GSAP SVG [online]. 2025 [cit. 2025-11-19]. Dostupné z: https://gsap.com/svg/ Obsah a relevance: Oficiální dokumentace knihovny GSAP se zamřením na SVG plugin.

LERSEN, Rob. Mastering SVG. Livery Place: Packt Publishing, 2018. ISBN 978-1-78862-674-3. Obsah a relevanci: propojení grafiky a kódu, jak správně strukturovat SVG, aby bylo animovatelné.

Modern SVG Animation Techniques. In: Open Replay [online]. 2025 [cit. 2025-12-20]. Dostupné z: https://blog.openreplay.com/modern-svg-animation-techniques/

OPENAI. ChatGPT [online]. 2025 [cit. 2025-11-23]. Dostupné z: https://chat.openai.com/

SHAPIRO, Julian. Web animation using JavaScript: Develop and Design. 2015. Peachpit Press, 2015. ISBN 978-0-134-09666-7.

Starý hrad [online]. 2025 [cit. 2025-11-19]. Dostupné z: https://www.staryhrad.cz/index.php/historie2/chronologicky2/do-18-stoleti

SVG, Canvas, WebGL, WebGPU — who is actually in charge here? A big test of 2D graphics in the browser. In: Habr [online]. 2025 [cit. 2025-12-20]. Dostupné z: https://habr.com/ru/articles/970286/

The Beginner's Guide to the GreenSock Animation Platform. FreeCodeCamp[online]. 2025 [cit. 2025-11-19]. Dostupné z: https://www.freecodecamp.org/news/the-beginners-guide-to-the-greensock-animation-platform-7dc9fd9eb826/

The processes of Malting Through the Ages. St Edmundsbury Chronicle [online]. 1997, 02.11.2025 [cit. 2025-11-23]. Dostupné z: http://www.stedmundsburychronicle.co.uk/maltsters/maltingprocess.htm

VOZISOV, Nikita, Ilya GOSUDAREV a Irina GOTSKAYA. Approaches towards the Comparison and Utilization of JavaScript Animation Libraries. CEUS [online]. 2019, 2590(2590), 1-9 [cit. 2025-11-19]. Dostupné z: https://ceur-ws.org/Vol-2590/short9.pdf