TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Webová aplikace pro generování realistického modelu kosti

Bakalářská práce

Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie

Autor práce: **Helena Doanová**
Vedoucí práce: doc. Ing. Petr Henyš, Ph.D.
Ústav nových technologií a aplikované informatiky

Liberec 2022

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Webová aplikace pro generování realistického modelu kosti

Jméno a příjmení: **Helena Doanová**
Osobní číslo: M18000070
Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2021/2022**

Zásady pro vypracování:

1. Seznamte se s tvorbou interaktivní webové aplikace a webové vizualizace v prostředí Panel a vtk.js. Dále prostudujte způsob generování modelu kosti s realistickými fyzikálními vlastnostmi v uvedené literatuře.
2. Na základě získaných znalostí navrhněte webovou aplikaci, která umožní uživateli vygenerovat výpočtový model kosti podle věku, pohlaví a přesnosti a umožní model uložit do standardních programů používaných pro inženýrské výpočty.
3. Aplikaci implementujte a ověřte její funkčnost porovnáním výstupu z aplikace s uvedenou literaturou a na ukázkovém příkladu výpočtu základních fyzikálních veličin kosti.

Rozsah grafických prací: dle potřeby dokumentace
Rozsah pracovní zprávy: 30-40 stran
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

[1] A high-level app and dashboarding solution for Python [online]. [cit. 9.10.2021]. Dostupné z: https://panel.holoviz.org/
[2] The Visualisation Toolkit (VTK) [online]. Kitware, 2021. [cit. 9.10.2021]. Dostupné z: https://vtk.org/
[3] Henyš, Petr et al. Bone mineral density modeling via random field: normality, stationarity, sex and age dependence. *Computer Methods and Programs in Biomedicine. 2021*, 210, 106353, ISSN: 0169-2607.

Vedoucí práce: doc. Ing. Petr Henyš, Ph.D.
Ústav nových technologií a aplikované informatiky

Datum zadání práce: 12. října 2021
Předpokládaný termín odevzdání: 16. května 2022

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 19. října 2021

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracovala samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědoma toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědoma povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědoma následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

2. května 2022 Helena Doanová

Webová aplikace pro generování realistického modelu kosti

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací webové aplikace pro generování realistického modelu kosti. Model kosti je reprezentován sítí tetrahedronů, které v sobě uchovávají kostní minerální hustotu. Matematický model pro výpočet kostní minerální hustoty bere ohledy na pohlaví i věk virtuálního pacienta. V teoretické části jsou popsány knihovny, které byly pro tvorbu aplikace použity. V praktické části je nejdříve představen návrh aplikace. Jsou zde popsány všechny požadavky na aplikaci a také její popis a architektura. Dále je popsána samotná implementace aplikace a její jednotlivé komponenty. V poslední části je ověřena funkčnost aplikace a matematického modelu.

Klíčová slova: Python, Panel, VTK, PyVista, kostní minerální hustota, virtuální pacient

Abstract

This bachelor thesis deals with the design and implementation of a web application for generating a realistic bone model. The bone model is represented by a network of tetrahedrons that retain bone mineral density. The mathematical model for calculating bone mineral density takes into account the sex and age of the virtual patient. The theoretical part describes the libraries that were used to create the application. The practical part first introduces the application design. All requirements for the application are described here, as well as its description and architecture. The implementation of the application and its individual components are also described. The last part verifies the functionality of the application and the mathematical model.

Keywords: Python, Panel, VTK, bone mineral density, virtual patient

Poděkování

Na tomto místě bych chtěla poděkovat vedoucímu mé bakalářské práce panu doc. Ing. Petru Henyšovi, Ph.D za odborné vedení, cenné rady a poznatky, které mi během zpracování této bakalářské práce poskytoval. Také bych ráda poděkovala za všechny konzultace, ochotu a trpělivost.

Obsah

Seznam zkratek 9

1 Úvod **11**
1.1 Matematický model 11

2 Použité knihovny **13**
2.1 Knihovna Panel 13
2.2 Knihovna VTK 14
2.3 Knihovna PyVista 15
2.4 Knihovna FEniCS 15
2.5 Server Bokeh 15
2.6 Knihovna ZeroMQ 17

3 Návrh webové aplikace **18**
3.1 Základní požadavky pro aplikaci 18
3.2 Popis aplikace 19
3.3 Architektura aplikace 19

4 Implementace aplikace **21**
4.1 Přehled komponent 21
4.2 Rozložení stránky 21
4.2.1 Popis rozložení aplikace 23
4.3 Nastavení parametrů pro výpočtový model 24
4.3.1 Nastavení pohlaví a věku 24
4.3.2 Možnost výběru kostí 25
4.3.3 Nastavení přesnosti výpočtového modelu 25
4.4 Vykreslování objektů v renderovacím okně 26
4.4.1 Objektová třída KLE 27
4.5 Spuštění výpočetního procesu 28
4.5.1 Komunikace mezi aplikací a výpočtovým serverem 29
4.5.2 Popis implementace matematického modelu 32
4.6 Renderovací okno 33
4.7 Loading spinner 35
4.8 Uložení výpočtového modelu kosti 36

5 Ověření funkčnosti aplikace **38**
5.1 Ověření závislosti kostní minerální hustoty na věku 38
5.2 Porovnání střední hodnoty a směrodatné odchylky 40

6 Závěr **42**

Literatura **44**

Přílohy **45**

A Zdrojové kódy **46**

Seznam zkratek

JSON	JavaScript Object Notation
KL	Karhunen-Loèveova
MPI	Message Passing Interface
OpenGL	Open Graphics Library
VTK	Visualization Toolkit

Seznam obrázků

2.1 Schéma panelu [3] . . . 14
2.2 Server Bokeh . . . 16

3.1 Architektura . . . 20

4.1 Oblasti šablony . . . 22
4.2 Rozložení aplikace . . . 23
4.3 Tabulka 1 . . . 24
4.4 Tabulka 2 [10] . . . 25
4.5 Tabulka 3 . . . 26
4.6 Tlačítko Start simulation . . . 29
4.7 Komunikace mezi aplikací a výpočtovým serverem . . . 31
4.8 Renderovací okno s více modely kostí . . . 33
4.9 Síť tetrahedronů . . . 34
4.10 Výběr barevné mapy . . . 35
4.11 Zobrazení kostí v barevné mapě *rainbow* a *Greys* . . . 35
4.12 Loading spinner . . . 36
4.13 Tlačítko Download . . . 36
4.14 Model kosti v ParaView . . . 37

5.1 Závislost kostní minerální hustoty na věku u žen . . . 39
5.2 Závislost kostní minerální hustoty na věku u mužů . . . 39
5.3 Histogram kostní minerální hustoty u žen . . . 40
5.4 Histogram kostní minerální hustoty u mužů . . . 41

1 Úvod

Cílem bakalářské práce je vytvořit interaktivní webovou aplikaci, která umožní uživateli vygenerovat výpočtový model kosti. Uživatel bude moci nastavit parametry jako je věk, pohlaví, typ kosti a přesnost. Model kosti bude obsahovat kostní minerální hustotu, což je důležitá vlastnost kosti. Dále aplikace umožní uložit model kosti do standardních programů používaných pro inženýrské výpočty.

V medicíně a obecně v lékařských vědách jsou experimenty extrémně zatížené vysokým rozptylem v datech, variabilitou pacientů a nejistotami. Klíčového množství vzorků je obtížné dosáhnout a proto se v poslední době prosazuje koncept digitálního pacienta ("digitální dvojče") [1], který by umožnil získat libovolné množství vzorků bez nákladných a zdlouhavých experimentů. Cílem bylo vytvořit interaktivní webovou aplikaci pro generování realistického modelu kosti, která posunuje možnosti využití virtuálního pacienta blíže ke klinické realitě. S touto aplikací je tedy možné generovat libovolné množství virtuálních kostí s minerální hustotou kosti respektující variabilitu populace a věkovou závislost. V této bakalářské práci virtuálního pacienta představuje webová aplikace pro generování modelu kosti s realistickým průběhem minerální hustoty. Výsledný výpočtový model kosti, který webová aplikace poskytuje, obsahuje informace o minerální hustotě kosti, její závislosti na věku a pohlaví a také průměrný tvar kosti.

Strukturální a vnitřní vlastnosti kosti jsou nehomogenní. Liší se napříč mnoha prostorovými a časovými měřítky a také populací. Kostní minerální hustota se běžně používá ke studiu vlastností kostí a také souvisí s elasticitou kostí a rizikem zlomenin. Zlomeniny pánve se obtížně léčí a stávající fixační postupy mají významnou míru selhání. Nižší hodnoty minerální hustoty jsou spojeny s vyšší pravděpodobností selhání fixace zlomeniny. Proto pánevní kost slouží jako dobrá volba k prokázání kostní minerální hustoty jako náhodného pole. [1]

1.1 Matematický model

Matematický model pro virtuálního pacienta funguje na principu modelace kostní minerální hustoty jako náhodného pole. K modelování minerální hustoty kosti jako náhodného pole s Gaussovými koeficienty byla použita *Karhunen-Loèveova expanze* (KL). [1] *Karhunen-Loèveova expanze* je také známá pod názvem *Analýza hlavních komponent*. KL expanze je transformace, která převádí korelované párové veličiny

na nekorelované veličiny. Ve studii byla tato transformace použita tak, že se minerální hustota nového pacienta vygenerovala pomocí nekorelovaných Gausovských proměnných, které byly následně pomocí zpětné KL expanze transformovány na korelované. [1] Celkový matematický předpis uvažovaný v [1] pro generování minerální hustoty pacienta je:

$$\boldsymbol{\rho}(\mathbf{x}) = \boldsymbol{\rho}_0(\mathbf{x}) + \boldsymbol{\rho}_1(\mathbf{x})t + \boldsymbol{\sigma}(\mathbf{x}) \sum_{i=1}^{P} \sqrt{\lambda_i}\theta_i \boldsymbol{\psi}_i(\mathbf{x}) \tag{1.1}$$

Kde $\boldsymbol{\rho}$ je minerální hustota v závislosti na poloze v kosti $\mathbf{x}$. $\boldsymbol{\rho}_0$ je funkce střední hodnoty v závislosti na poloze v kosti. $\boldsymbol{\rho}_1$ je funkce koeficientu vyjádření lineární závislosti minerální hustoty na věku t. $\boldsymbol{\sigma}(\mathbf{x})$ je funkce směrodatné odchylky. Dvojice λ_i a $\boldsymbol{\psi}_i(\mathbf{x})$ jsou vlastní čísla a vlastní vektory korelační matice použití v KL expanzi. θ_i je nekorelovaná Gausovská veličina s jednotkovou směrodatnou odchylkou a nulovou střední hodnotou. Matematický model zohledňuje rozdíly mezi pohlavími, a proto je nutné, aby dataset byl rozdělen do dvou souborů na základě pohlaví. K vyjádření vztahu mezi věkem a kostní minerální hustotou je použita lineární regrese. Číslo P udává, kolik se má použít v KL expanzi vlastních vektorů a čísel. Pokud se zahrnou všechna vypočítaná vlastní čísla a vektory korelační matice, pak je model nejpřesnější. Obecně zpravidla stačí prvních několik největších vlastních čísel a vektorů k získání dostatečné přesnosti modelu.

2 Použité knihovny

V této kapitole jsou stručně popsány knihovny, které byly použity k tvorbě aplikace a k ověření její funkčnosti.

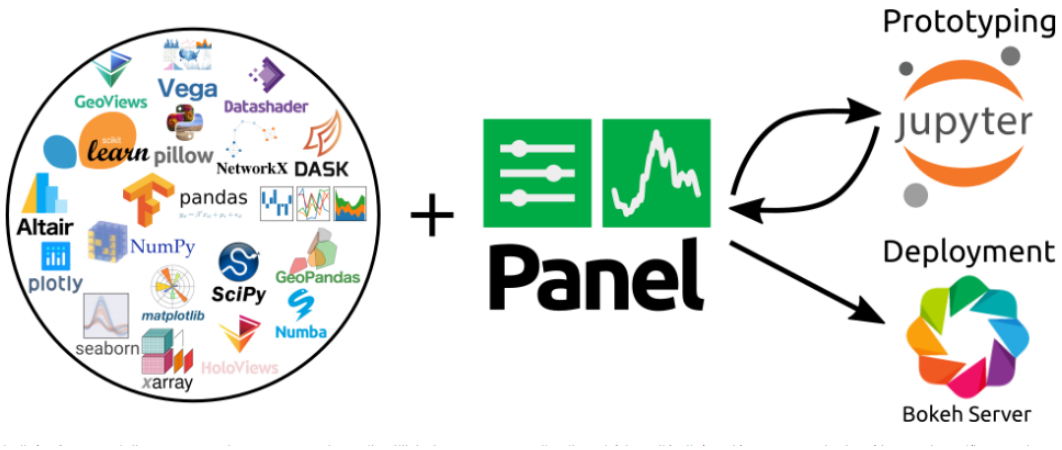
2.1 Knihovna Panel

Panel je volně dostupná knihovna Pythonu, která umožňuje vytvářet vlastní interaktivní webové aplikace a řídící panely. Uživatelem definované widgety propojuje s grafy, obrázky, tabulkami nebo textem. [2]

Hlavní myšlenkou *Panel* je co nejsnadněji zabalit výstupy existujících nástrojů v ekosystému PyData do ovládacího panelu, aplikace nebo řídícího panelu. Tímto se zajistí, že uživatelé mohou bezproblémově pracovat s analytickými a vizualizačními nástroji které, již znají. Usnadňuje také přechod od prototypování malé aplikace k nasazení nebo veřejnému sdílení se zbytkem světa. [3] Schéma navržení *Panel* je vidět na obrázku 2.1.

Velkou výhodou knihovny *Panel* je, že podporuje téměř všechny vykreslovací knihovny Pythonu. Vizualizace lze tím pádem okamžitě zobrazit. A to buď samostatně, nebo v kombinaci s interaktivními widgety, které je ovládají. Funguje stejně dobře jak v Jupiter Notebook, tak i na samostatném zabezpečeném serveru. V obou případech používá stejný kód, což umožňuje snadné přepínání mezi prozkoumáváním dat, vytvářením vizualizací, přidáváním vlastní interaktivity, sdílení s netechnickými uživateli a zpět kdykoli pomocí stejných nástrojů. *Panel* podporuje prostředí využívající Python backend, ale je také možné exportovat aplikaci jako statické HTML/Javascript. Další výhodou je nezávaznost kódu na konkrétním GUI nebo webových nástrojích. Také umožňuje snadné vytvoření a práci s Python webovými servery podporující velké výpočty dat. *Panel* je postaven na knihovně *Bokeh*, která poskytuje výkonný webový server. [2, 4]

Objekty knihovny *Panel* jsou reaktivní. Okamžitě se aktualizují, aby odpovídaly stavu ve kterém jsou. To usnadňuje sestavování zobrazitelných objektů a jejich propojení do aplikací. Stejné objekty pak mohou být znovu použity ve složitějších kombinacích, přičemž vždy sdílí stejný kód. [4]



Obrázek 2.1: Schéma panelu [3]

2.2 Knihovna VTK

VTK je volně dostupný softwarový systém pro 3D počítačovou grafiku, modelování, zpracování obrazu, objemové vykreslování, vědeckou vizualizaci a 2D vykreslování. Podporuje širokou škálu vizualizačních algoritmů a pokročilých modelovacích technik. Využívá výhody paralelního zpracování ve vláknech i distribuované paměti pro rychlost a škálovatelnost. [5]

VTK může být použito téměř na jakékoli platformě (Linux, Windows, Mac, na webu, na mobilních zařízeních). *VTK* staví, testuje a balí systém pomocí kvalitního softwarového procesu Kitware, který zahrnuje CMake, CTest, CDash a CPack. Pro maximalizování efektivity je základní funkcionalita *VTK* napsána v C++. Aby mohla být tato funkcionalita použita širším publikem, je zabalena do jiných jazykových vazeb. Obzvlášť dobře zpracovaná je interoperabilita s Pythonem. *VTK* má řadu užitečných vlastností: [5]

- *VTK* aplikace manipuluje s daty pomocí filtrů. Každý filtr kontroluje data která přijímá a vytváří odvozená data. Propojená sada filtrů tvoří síť toku dat. Nezpracovaná data jsou pomocí konfigurovatelné sítě transformována do vizuálně srozumitelnějších formátů.
- *VTK* přidává vykreslovací abstrakční vrstvu přes základní grafickou knihovnu (většinou OpenGL). Tato vyšší úroveň zjednodušuje vytvoření přesvědčivé vizualizace.
- Základní datový model *VTK* je schopen reprezentovat téměř každý reálný problém související s fyzikální vědou. Pro lékařské zobrazování a inženýrskou práci, která zahrnuje řešení konečných rozdílů a konečných prvků, jsou zvláště vhodné základní datové struktury.

- Datová interakce napomáhá porozumět obsahu, tvaru a významu dat. *VTK* je také vysoce kompatibilní s Pythonem, včetně Matplotlib.
- *VTK* poskytuje silnou podporu pro škálovatelné paralelní zpracování distribuované paměti založené na MPI.

2.3 Knihovna PyVista

PyVista je pomocný modul a vysokoúrovňové rozhraní pro *VTK*, který práci s touto knihovnou ulehčuje. S *VTK* je propojený prostřednictvím *NumPy* a přímého přístupu k poli. Umožňuje využívat síťové datové struktury a metody filtrování pro prostorové datové sady. Užitečné je 3D vykreslování, které je jednoduché a vytvořené pro velké či složité datové geometrie. Dále poskytuje rozhraní Pythonic, které je dobře zdokumentované a odhaluje výkonný vizualizační backend *VTK*. Díky tomu se usnadní rychlé prototypování, analýza a vizuální integrace prostorově odkazovaných datových sad. [6]

2.4 Knihovna FEniCS

FEniCS je velmi oblíbená volně dostupná výpočetní platforma pro řešení parciálních diferenciálních rovnic a numerických úloh. *FEniCS* umožňuje uživatelům rychle převádět vědecké modely do efektivního kódu konečných prvků. S vysokoúrovňovým rozhraním Pythonu a C++ je použití *FEniCS* vcelku jednoduché. *FEniCS* ale také nabízí výkonné možnosti pro více zkušenné programátory. Každá součást platformy *FEniCS* byla navržena s ohledem na paralelní zpracování. Tento framework umožňuje rychlé prototypování formulací konečných prvků a řešitelů na laptopech a pracovních stanicích. Stejný kód může být použit i na velkých vysoce výkonných počítačích. [7] *FEniCS* běží na mnoha platformách od notebooků po vysoce výkonné clustery. [7] V této práci bude tato knihovna použita jenom pro výpočet hmotnosti kosti, tj. integrálu:

$$w = \int_{\Omega} \boldsymbol{\rho}(\mathbf{x})\,\mathrm{d}\mathbf{x} \tag{2.1}$$

Tento integrál nelze spočítat analyticky a protože je geometrie kosti reprezentována jako množina tetrahedrů, lze spočítat hmotnost každého tetrahedru a pak je sečíst přes celou množinu a získat tak tím celkovou hmotnost kosti v závislosti na hustotě.

2.5 Server Bokeh

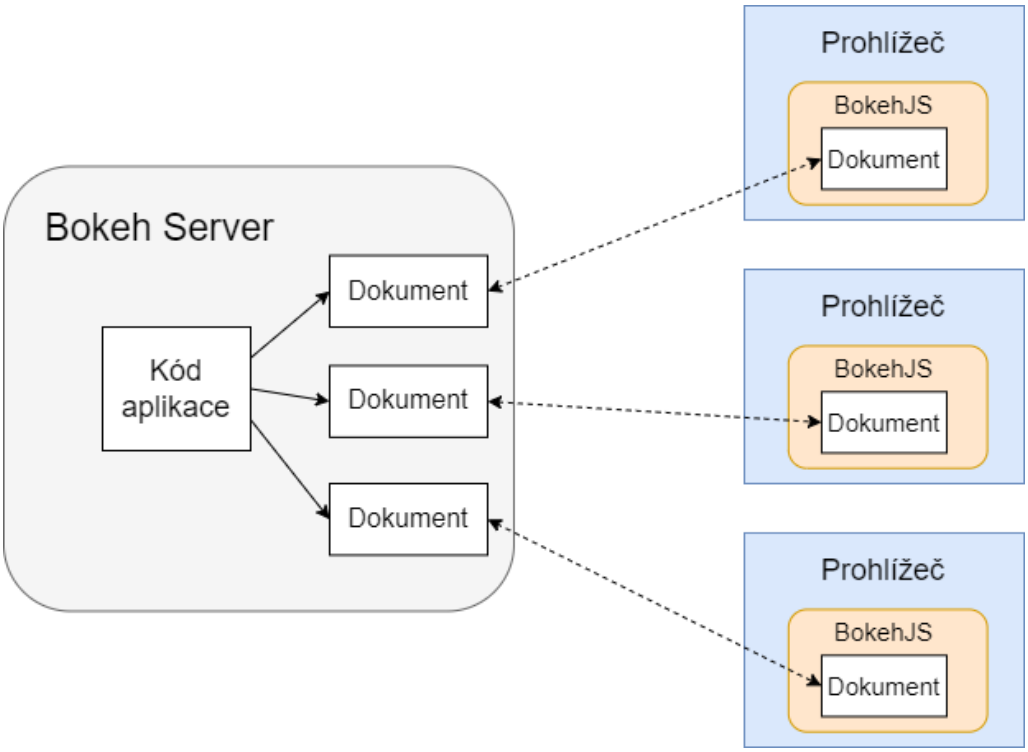
Bokeh server usnadňuje vytváření interkativních webových aplikací, které propojují front-endové uživatelské rozhraní se spuštěným Python kódem. *Bokeh* vytváří vysokoúrovňové Python modely jako jsou grafy, rozsahy, osy a glyfy. Poté tyto objekty převádí na JSON, aby je předal své klientské knihovně *BokehJS*. [8]

Tento flexibilní a oddělený design má své výhody. Například je snadné mít v prohlížeči jiné jazyky, jako je R nebo Scala, nebo řídit vykreslování a vizualizace *Bokeh*. Nicméně udržování těchto modelů v synchronizaci mezi prostředím Pythonu a prohlížečem poskytuje více výkonných funkce: [8]

- Umožňuje reagovat na události uživatelského rozhraní generované v prohlížeči pomocí výpočtů nebo dotazů s plnou silou Pythonu.
- Automaticky posílá aktualizace na straně serveru do prvků uživatelského rozhraní jako jsou widgety nebo grafy v prohlížeči.
- Používá periodická, časově omezená a asynchronní zpětná volání k zajištění aktualizací streamování.

Hlavím úkolem serveru *Bokeh* je udržovat data synchronizovaná mezi základním prostředím Pythonu a knihovnou *BokehJS* spuštěnou v prohlížeči. Změny ovládacích prvků uživatelského rozhraní jsou sdělovány backendu prostřednictvím serveru *Bokeh*. To také spouští zpětná volání, která aktualizují grafy pomocí vstupu v reálném čase. [8]

Server *Bokeh* použije kód aplikace k vytvoření relací a dokumentů pro všechny připojené prohlížeče. [8] Tento proces je znázorněn na obrázku 2.2.



Obrázek 2.2: Server Bokeh

S každým novým připojením (vpravo) *Bokeh* server (vlevo) spustí kód aplikace a vytvoří nový *Bokeh* dokument, který synchronizuje s prohlížečem. Kód aplikace také vytváří zpětná volání, která by se měla spustit vždy, když se změní vlastnosti jako jsou hodnoty widgetu. [8]

2.6 Knihovna ZeroMQ

ZeroMQ je velmi výkonná asynchronní knihovna pracující se zprávami. Je navržena pro použití v distribuovaných nebo souběžných aplikacích. Poskytuje frontu zpráv, ale na rozdíl od middlewaru orientovaného na zprávy nevyžaduje systém *ZeroMQ* vyhrazeného zprostředkovatele zpráv. [9]

ZeroMQ podporuje běžné vzorce pro zasílání zpráv jako jsou pub/sub, request/replay, client/server a další. K tomu využívá různé přenosy jako TCP, in--process, inter-process, multicast, WebSocke a další. Díky tomu je zasílání zpráv mezi procesy stejně jednoduché jako zasílání zpráv mezi vlákny. Zdrojový kód tím pádem bude čitelnější, modulárnější a škálovatelnější. [9]

Hlavní myšlenka, která se skrývá za *ZeroMQ* je číslo nula. Nula znamená žádný broker, nulová latence, nulové náklady a nulová administrace. [9]

3 Návrh webové aplikace

Cílem bylo vytvořit interaktivní webovou aplikaci pro generování realistického modelu kosti, která by měla vyřešit problém s nedostatečným počtem vzorků v biomedicínském inženýrství. Tato kapitola je zaměřena na požadavky, popis a architekturu aplikace.

3.1 Základní požadavky pro aplikaci

Před i v průběhu tvorby webové aplikace byly sestavovány požadavky, které buď byly zásadní pro správný chod aplikace nebo měly zpříjemnit uživatelský zážitek. Těmito požadavky byly:

- Možnost, aby uživatel mohl vygenerovat výpočtový model kosti na základě zadaných parametrů. Tedy aby uživatel mohl nastavit věk, pohlaví a přesnost virtuálního pacienta. Dalším parametrem bylo nastavení typů kosti, které chce uživatel generovat.
- Aplikace by měla také umožnit uložit výpočtový model kosti do standardních programů používaných pro inženýrské výpočty.
- Jelikož je výpočet minerální hustoty kosti časově náročnější, měla by být aplikace plně interaktivní a schopná informovat uživatele, že se výpočet stále provádí. K tomuto účelu skvěle slouží funkce loading spinneru.
- Aplikace by měla mít možnost zobrazit výpočtový model kosti vedle standardní barevné mapy ”rainbow” také v odstínech šedi.
- Dále by aplikace měla uživatele, který nemá zkušenosti s knihovnou *VTK*, informovat, jak si práci s renderovacím oknem usnadnit.
- Hlavním prvkem webové aplikace by mělo být samotné renderovací okno. Nastavování parametrů by mělo být přehledné a jednoduché, jelikož tyto parametry bude uživatel často měnit.
- Na webové stránce by se také měl nacházet odkaz na studii, ze které matematický model vychází. Dále by se zde měla nacházet informace o autorovi webové aplikace a jeho email pro případ, kdyby uživatel měl nějaké dotazy.

Také by se zde mělo nacházet varování ohledně možného použití webové aplikace k lékařské diagnóze, jelikož tato aplikace rozhodně neslouží k tomuto účelu.

3.2 Popis aplikace

Aplikace byla tvořena v prostředí *Panel* a psána v jazyce Python. Prostředí *Panel* bylo zvoleno proto, že podporuje téměř všechny vykreslovací knihovny Pythonu. Právě tento aspekt byl při výběru důležitý, jelikož aplikace pracuje s vykreslovací knihovnou *VTK*. Knihovnu *VTK* aplikace využívá k zobrazení renderovacího okna a k renderování výpočtového modelu kosti a minerální hustoty.

Dalším důvodem, proč bylo zvoleno prostředí *Panel* je to, že je postaveno na knihovně *Bokeh*, která poskytuje výkonný webový server pro komunikaci mezi Pythonem a prohlížečem. Tento webový server zvládá i náročnější výpočetní operace. To se hodí zejména při renderování a vykreslování, které probíhá přímo v prohlížeči. Aplikaci bylo dáno jméno *BoneGen*.

3.3 Architektura aplikace

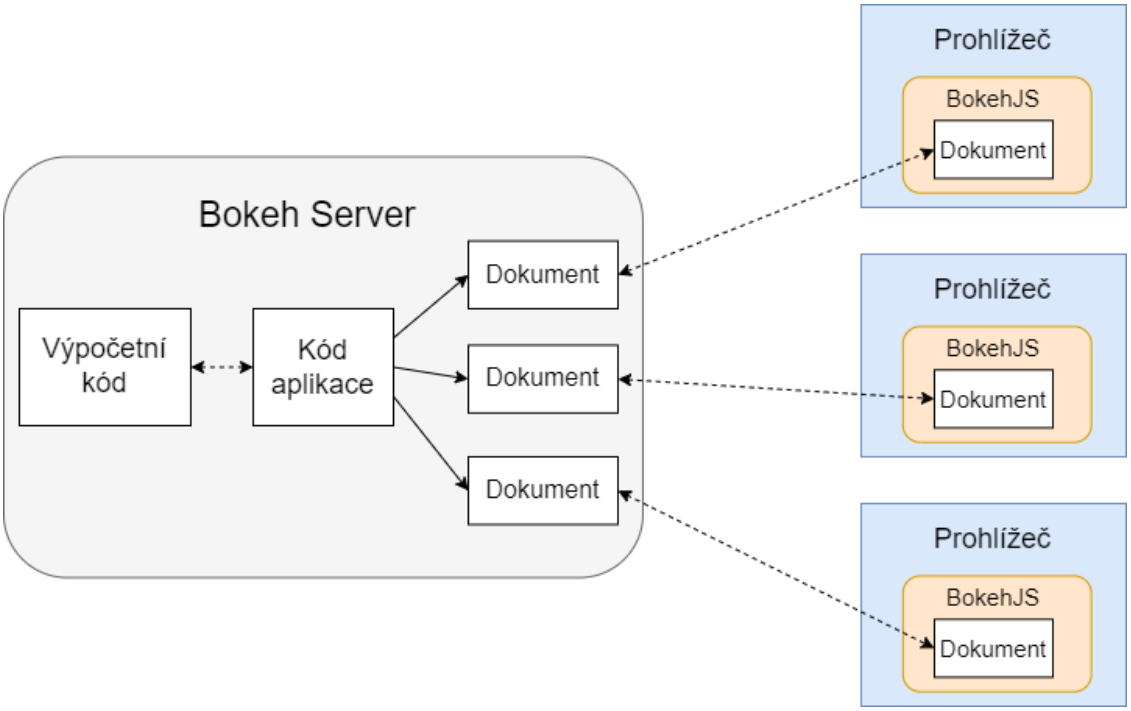
Jak je již zmíněno výše, prostředí *Panel* poskytuje vlastní webový server *Bokeh*. Server *Bokeh* se stará o komunikaci mezi Pythonem a prohlížečem. Jak tato komunikace funguje je popsáno v sekci 2.5.

Aplikace se skládá ze dvou python skriptů, které jsou uloženy na serveru. Propojeny jsou knihovnou *ZeroMQ*, s jejíž pomocí je vytvořen kanál mezi těmito dvěma skripty.

První skript (kód aplikace) obsahuje logiku a backend aplikace. Stará se o rozložení webové stránky a propojení jednotlivých komponentů s funkcemi. Druhý skript (Výpočetní kód) obsluhuje načítaní dat a matematický model. Stará se o načtení datasetů a následné uložení dat tak, aby s nimi bylo možné nadále pracovat. Dále provádí samotný výpočet minerální hustoty kosti.

Pro takovéto řešení, v podobě rozdělení aplikace do dvou samostatných skriptů, bylo rozhodnuto na základě:

- Oddělení výpočtové části od zbytku kódu. Tímto rozdělením se dosáhne přehledného a jednoduše rozšiřitelného zdrojového kódu.
- Z důvodu zkrácení čekací doby pro uživatele. Samotné načtení datasetů značnou chvíli trvá, jelikož data jedné kosti jsou tvořena z pole floatů o velikosti přibližně milion. Takto na *Bokeh* serveru poběží ještě menší server (Výpočetní



Obrázek 3.1: Architektura

kód), na kterém již data načtená budou. Klient (Kód aplikace) mu bude pomocí knihovny *ZeroMQ* zasílat jen dotazy a vyhne se tím celkem dlouhému procesu načítání datasetů. Po přijetí a zpracování dotazu, server určený pro výpočet (Výpočetní kód) zašle klientovi zpátky výpočtový model kosti.

4 Implementace aplikace

Aby bylo možné porozumět jak jednotlivé komponenty fungují, nachází se v této kapitole jejich přehled. Dále je zde popsáno, k jakému účelu se jednotlivé komponenty aplikace používají a kde se nachází. Také jsou zde popsány hlavní funkcionality aplikace a komunikace se serverem.

4.1 Přehled komponent

Panel obsahuje různé komponenty pro rychlé sestavení panelů, aplikací a řídících panelů. Jsou zde tři typy tříd objektů, které tvoří komponenty:

- `Pane` – Pomocí panelových objektů lze zobrazit a uspořádat na stránce například grafy(Matplotlib, Plotly, HoloViews), fotografie(PNG,SVG, GIF) a různé značkovací jazyky(Markdown, LaTeX, HTML) .
- `Widget` – Jsou to řídící objekty, které mohou spouštět události Pythonu nebo Javascriptu. Umožňují uživateli vkládat data do aplikace nebo řídícího panelu.
- `Panel` – Je hierarchický kontajner, který lze použít k uspořádání více komponent (pane,widget, nebo jiných panelů) do aplikace nebo řídících panelů. Existují čtyři hlavní typy panelů(řádaek, sloupec, tabulka a gridspec).

Všechny objekty používají stejné rozhraní API, což usnadňuje změnu jejich chování při rozvržení, vizuální vzhled a také jejich propojení, zobrazení a export.

Všechny komponenty v *Panel* jsou postaveny na knihovně Param. Každá komponenta deklaruje sadu parametrů, které řídí její chování a výstup. Základním principem však je, že hodnoty parametrů lze ovládat na úrovni třídy i na úrovni jednotlivých instancí.

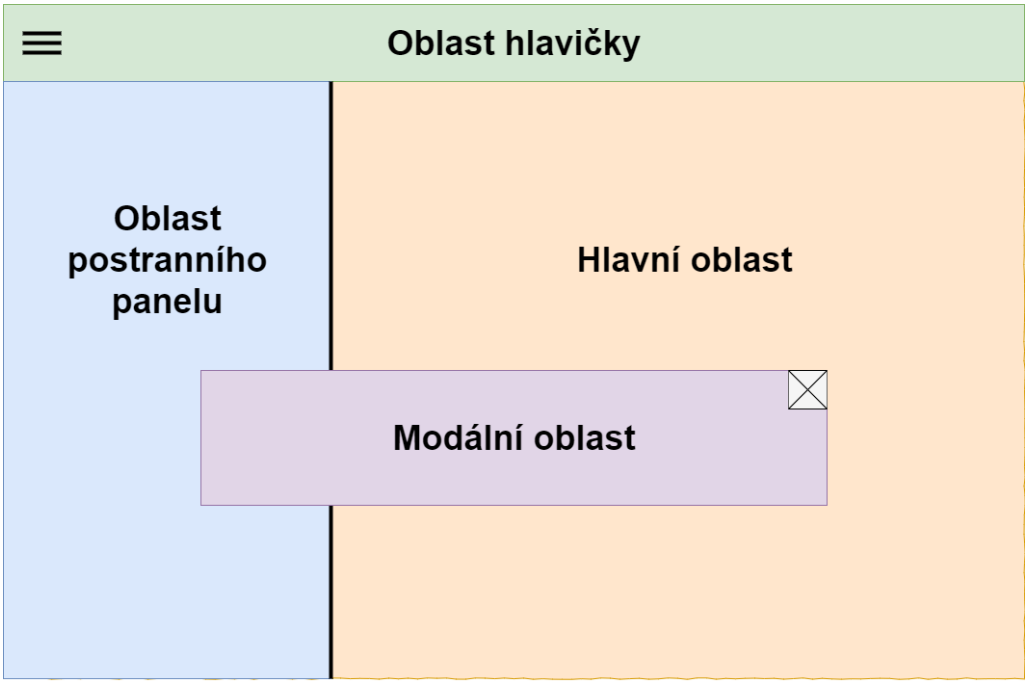
4.2 Rozložení stránky

Prostředí *Panel* vykresluje aplikace, které jsou v něm tvořené, do své výchozí šablony. Často je však požadováno přizpůsobení rozvržení aplikace nebo dokonce vložení více nezávislých panelů do aplikace. Komponenta Template v *Panel* umožňuje přizpůsobit výchozí šablonu a také vykreslit mnoho komponent na jedné stránce.

K definování šablon se používá šablonovací jazyk Jinja2. To usnadňuje rozšíření nebo dokonce celkovou změnu výchozí šablony. *Panel* obsahuje řadu šablon pro různé scénáře, z nichž všechny jsou založeny na různých CSS knihovnách. Jelikož při tvorbě aplikace nebylo třeba mít úplnou kontrolu nad přesným rozložením každé jednotlivé komponenty na stránce, byla zvolena jedena z možných šablon, které *Panel* nabízí. Zvolená šablona se nazývá BootstrapTemplate. Tato šablona je postavena na frameworku Bootstrap v4. Pro použití šablony BootstrapTemplate bylo rozhodnuto právě kvůli frameworku, na kterém je postavena. Framework Bootstrap je celosvětově nejoblíbenější a nejpoužívanější front-endová sada nástrojů s otevřeným zdrojovým kódem.

Šablona BootstrapTemplate je definována určením čtyř primárních oblastí obsahu na stránce. Tak, jak je vidět na obrázku 4.1. Tyto oblasti lze naplnit podle vlastní potřeby:

- `Oblast hlavičky` – Oblast záhlaví stránky.
- `Oblast postranního panelu` – Postranní panel, který lze sbalit.
- `Hlavní oblast` – Hlavní oblast aplikace.
- `Modalní oblast` – Modální oblast řízená Pythonem, která lze otevřít a zavřít.



Obrázek 4.1: Oblasti šablony

Šablonu lze použít k vývoji aplikace jedním ze dvou způsobů. Buď explicitně šablonu vytvořit, nebo změnit globální šablonu. Pro vytvořenou aplikaci byl použit explicitní přístup, kdy byla šablona vytvořena přímo, a poté do různých jejích částí byly přidávány komponenty.

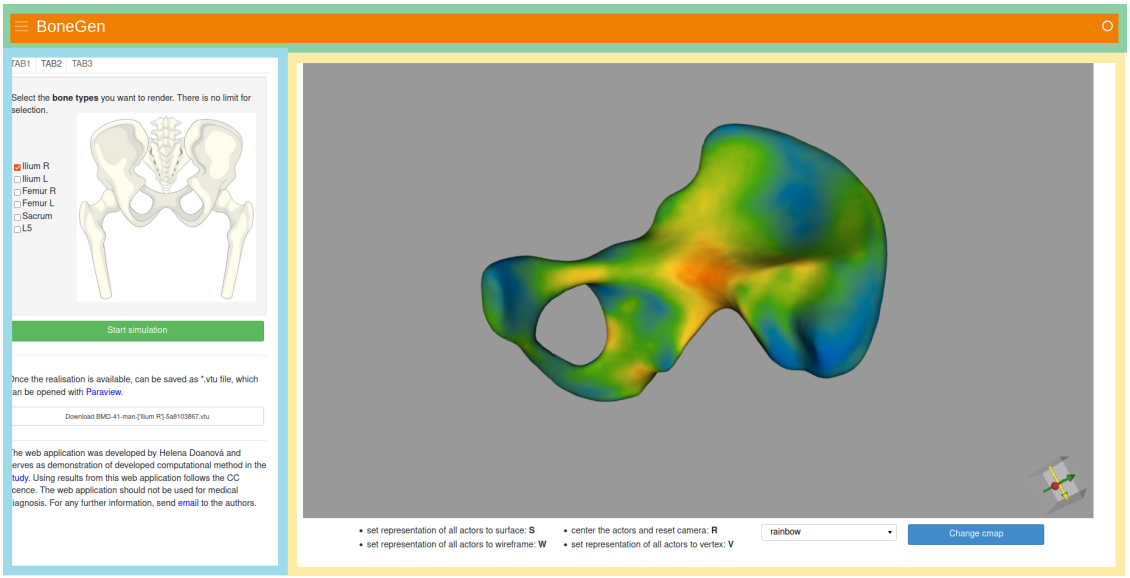
4.2.1 Popis rozložení aplikace

Celá vytvořená aplikace je rozdělena a zobrazena do čtyř výše uvedených oblastí, jak je vidět na obrázku 4.2 . V oblasti hlavičky se nachází hamburgerové tlačítko, které umožňuje sbalení a následné rozbalení postranního panelu. Vedle něj se zobrazuje název aplikace. Úplně vpravo se pak nachází vizuální indikátor, který znázorňuje zaneprázdněný stav aplikace.

V oblasti postranního panelu se nachází panel, který obsahuje 3 tabulky. Mezi tabulkami lze libovolně překlikávat. Tabulky obsahují widgety pro zadání parametrů, které jsou důležité pro generování modelu kosti. Pod tabulkami je tlačítko *Start simulation*, které spustí výpočet pro generaci modelu kosti. Pod tímto tlačítkem se nachází ještě jedno tlačítko, kterým tento model kosti budeme moci stáhnout. Dále se v postranním panelu nachází informace o studii,ze které výpočetní model pochází, kontakt na autora aplikace a také varování, že by tato aplikace neměla sloužit k léčebné diagnóze.

Hlavní obsah stránky tvoří renderovací okno. Pod ním se nachází informace o klávesových zkratkách, které slouží k jeho obsluze. Dále se zde nachází možnost volby, v jaké barevné mapě se má výpočtový model kosti zobrazit.

Modální oblast v aplikaci nebyla nijak využita, jelikož je žádoucí, aby většina komponent byla uživateli stále na očích.



Obrázek 4.2: Rozložení aplikace

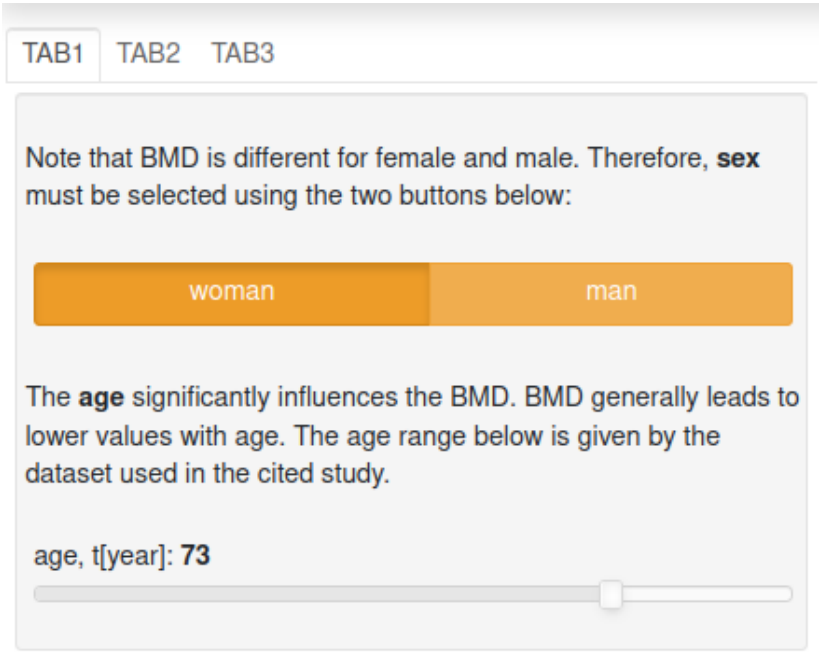
4.3 Nastavení parametrů pro výpočtový model

Aby matematický model pro výpočet minerální hustoty kosti správně fungoval a byl co nejpřesnější, je nutné mu zadat parametry. Jednou z hlavních podmínek pro vytvořenou webovou aplikaci bylo právě umožnit uživateli tyto parametry nastavit.

Nastavení parametrů se nachází v levé části stránky v postranním panelu. Zde se nachází další panel, který obsahuje tři tabulky. Tyto tabulky obsahují další komponenty včetně widgetů pro nastavení parametrů. Uživatel má možnost nastavit pohlaví a věk virtuálního pacienta. Dále má možnost vybrat, jaké kosti chce nechat vykreslit. Uživatel také může nastavit přesnost výpočtového modelu kosti. Každý parametr má již přednastavenou výchozí hodnotu. Na základě těchto nastavených hodnot se mění minerální hustota výpočtového modelu kosti.

4.3.1 Nastavení pohlaví a věku

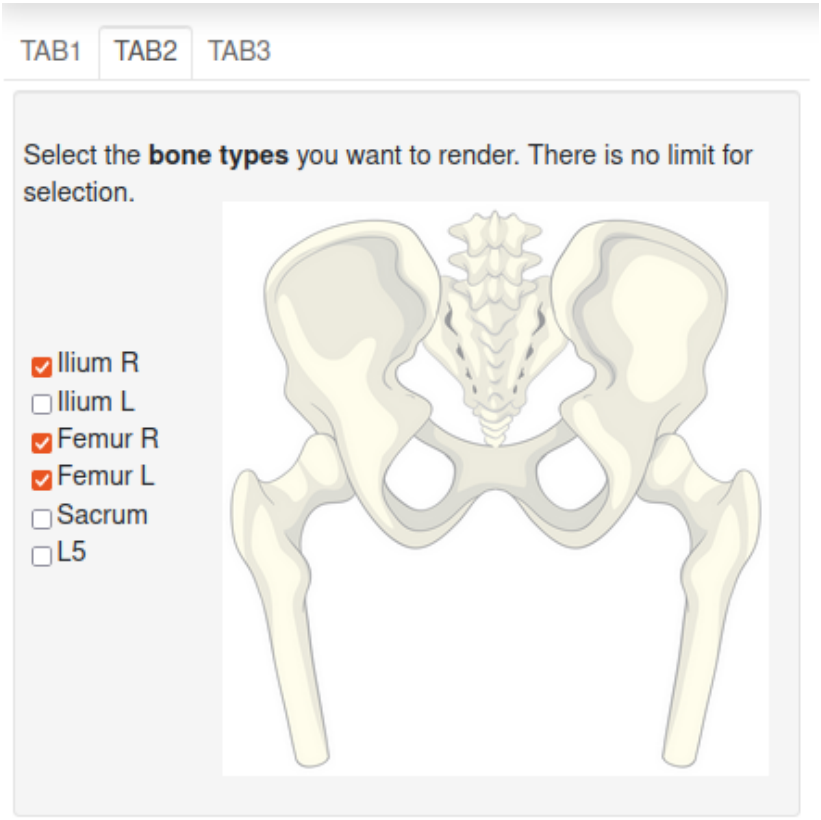
V první tabulce se nachází skupina dvou tlačítek pro výběr pohlaví a slider pro nastavení věku virtuálního pacienta. Věkový rozsah, který slider nabízí, se pohybuje v rozmezí od 22 do 89 let. Právě tyto věkové kategorie jsou obsaženy v použitém datasetu. [1]



Obrázek 4.3: Tabulka 1

4.3.2 Možnost výběru kostí

Ve druhé tabulce se nachází skupina checkbox tlačítek, která dává uživateli možnost vybrat si, které kosti chce nechat vykreslit. Výběr není nijak omezený. Lze zaškrtnout jen jednu kost nebo klidně všechny kosti.



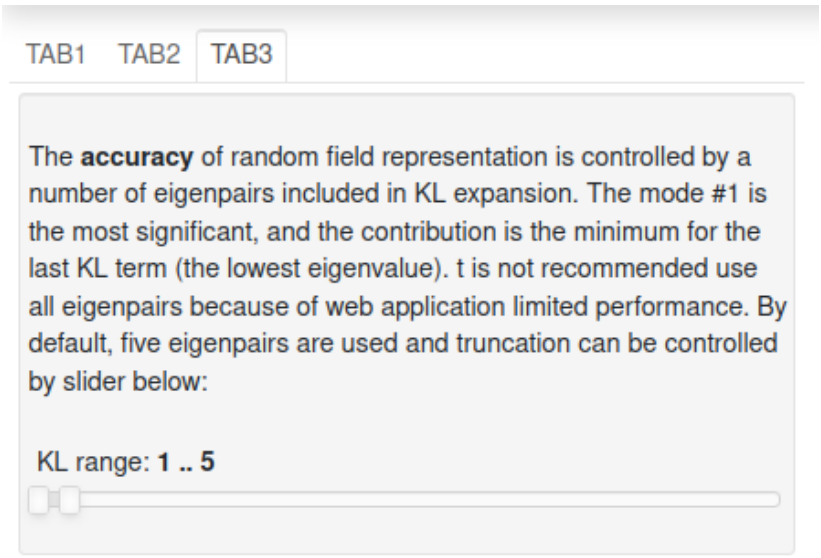
Obrázek 4.4: Tabulka 2 [10]

4.3.3 Nastavení přesnosti výpočtového modelu

Ve třetí tabulce se nachází slider pro nastavení přesnosti výpočtového modelu. V matematickém modelu tímto parametrem nastavujeme přesnost náhodného pole (parametr P, který může mít dolní a horní mez), které reprezentuje minerální hustotu kosti.

Rozsah slideru se mění podle toho, jaké pohlaví uživatel zvolil. To je dáno tím, že datasety pro matematický model vychází z reálných dat počítačové tomografie. Tím pádem záleží, kolik vzorků dat pro jednotlivé pohlaví datasety obsahují. Datasety se kterými aplikace pracuje obsahují data počítačové tomografie 97 žen a 88 mužů. Když je zvolena žena, rozsah slideru je v rozmezí od 1 do 97. Když je zvolen muž, rozsah slideru se změní na rozmezí od 1 do 88. [1]

Zahrnutím všech vlastních čísel a vektorů (rozsah 1-98) dostaneme nejpřesnější model, ale náročnější na výpočet. [1] Proto obvykle používáme prvních pět až deset vlastních vektorů, se kterými je matematický model dostatečně přesný a zároveň nenáročný na výpočet. Tím pádem kdybychom vybrali vlastní vektory například mezi 20-30, tak přesnost výpočtového modelu nebude dostatečná. Je nutné tedy zahrnout prvních pět vlastních vektorů.



Obrázek 4.5: Tabulka 3

4.4 Vykreslování objektů v renderovacím okně

V hlavním obsahu stránky se nachází renderovací okno. Aby aplikace toto okno mohla zobrazit a interaktivně s ním pracovat, byly použity knihovny *PyVista* a *VTK*. Knihovna *PyVista* poskytuje třídu *Plotter*, která umožňuje vytvořit renderovací okno. Třída *Plotter* je vykreslovací objekt pro zobrazování *vtk sítí* nebo *numpy* polí. Díky propojení knihovny *PyVista* a knihovny *Panel* lze renderovací okno interaktivně ovládat.

Kolem renderovacího okna je vytvořen *panel vtk*. Panel vtk se stará o vykreslení VTK scény uvnitř panelu, což umožňuje interakci se složitými geometriemi ve 3D. Také umožňuje synchronizaci stavu mezi *vtkRenderWindow* definovaným na straně pythonu a stavem prezentovaným v podokně prostřednictvým *vtk-js*. Python v tomto případě slouží jako server, který klientovi předává infromace o scéně. Synchronizace se zde provádí jen jedním směrem a to z Pythonu do JavaSriptu. Změny provedené na straně Javascriptu nejsou promítnuty zpět do *vtkRenderWindow*.

4.4.1 Objektová třída KLE

V kódu vytvořené aplikace se nachází objektová třída *KLE*. Tato třída slouží k vytvoření objektu, který obsahuje plotter s vtk panelem. Je zde umístěno také načítání vtk sítí, které se provádí, jakmile je aplikace spuštěna. Načtená síť je datová sada nestrukturované mřížky, která se skládá z jakéhokoli možného typu buněk v libovolné kombinaci. Nestrukturované mřížky jsou definovány body, buňkami a typy buněk. Tyto sítě reprezentují geometrii kosti průměrného pacienta (kolekce tetrahedrů).

Třída *KLE* dále zařizuje přidávání načtených vtk sítí do plotteru. Poté co jsou sítě přidány do plotteru stávají se z nich aktéři scény, se kterými lze dále pracovat prostřednictvím jejich parametrů. Aktéři ve *VTK* nejsou indexovány a proto se k nim v aplikaci přistupuje skrze *vtkActorCollection*. Ve třídě jsou definovány dvě tyto kolekce, protože uživatel má na výběr ze dvou možností barevných map. Barevná mapa se přidává přímo vtk sítím a nikoli aktérům, proto je nutné přidat sítě kostí dvakrát. Pokaždé s jinou zadanou barevnou mapou. Jedna kolekce obsahuje síť s barevnou mapou *rainbow* a druhá obsahuje síť s barevnou mapou *Greys*. Takto vzniknou dva aktéři pro jednu kost. Každý z nich se bude zobrazovat v jiné barevné mapě. Tyto aktéři se navzájem překrývají, takže když uživatel přepne barevnou mapu, kost pro něj zůstane na stejném místě. Vždy je zobrazen pouze jeden z těchto aktérů a druhý zůstává neviditelný.

Všem aktérům je ze začátku vypnuta viditelnost. Viditelnost se vybraným aktérům zapne až poté, co je zmáčknuto tlačítko *Start simulation* a provede se výpočetní proces popsaný v sekci 4.5.

```
class KLE(object):
    def __init__(self, plotter_surface):
        """Inicializace třídy"""

                    ...

    def load_meshes(self):
        """Načtení vtk sítí"""
        for bone_type in self.bone_type:
            self.meshes_rainbow[bone_type] = pv.read(bone_type +
            "/average_patient.xml")
        self.meshes_greys = copy.deepcopy(self.meshes_rainbow)

    def add_meshes_to_plotter_surface(self):
        """Přidání sítí do plotteru i s posunem(translate)
        Pro barevnou mapu rainbow i Greys """
        li.start_loading_spinner(model.vtkpan)
        for i, bone_type in enumerate(self.bone_type):
            KLE.update_meshes(self, self.meshes_rainbow[bone_type])
```

```
            mesh_actor_rainbow=self.plotter_surface.add_mesh(
            self.meshes_rainbow[bone_type].translate(100*i,100,100),
            cmap="rainbow")
            mesh_actor_rainbow.VisibilityOff()
            self.actor_collection_rainbow.AddItem(mesh_actor_rainbow)

            """ To samé se provádí pro barevnou mapu Greys ..."""
        li.stop_loading_spinner(model.vtkpan)

                    ...

    def create_vtkpanel(self):
        """Vytvoření vtk panelu"""
        self.vtkpan = pn.panel(self.plotter_surface.ren_win,
        orientation_widget=True, enable_keybindings=True,
        width=1450, height=850)
                    ...
```

4.5 Spuštění výpočetního procesu

Pro zahájení výpočetního procesu minerální hustoty kosti a následného vykreslení výpočtového modelu kosti, je nutné zmáčknout tlačítko *Start simulation*, které je vidět na obrázku 4.6. Na toto tlačítko je navázána metoda *draw_realisation*. Ta nejdříve zkontroluje jakou hodnotu obsahuje widget pro výběr barevné mapy, a poté jaké všechny typy kostí uživatel zaškrtl pro vykreslení. Dle toho se nastaví viditelnost jednotlivým aktérům v plotteru.

Pro každou jednotlivou zaškrtnutou kost metoda zjišťuje, jaké hodnoty parametrů pro výpočtový model uživatel nastavil. Tedy jaké hodnoty jsou obsaženy ve widgetech pro nastavení pohlaví, věku a přesnosti výpočtového modelu. Typ aktuálně dané kosti a získané hodnoty jsou následně uloženy do slovníku a odeslány za pomoci knihovny *ZeroMQ* výpočtovému serveru v podobě dotazu. Hlouběji popsaná komunikace mezi aplikací a výpočtovým serverem se nachází v podkapitole 4.5.1. Poté co se dotaz zpracuje a provede se výpočet minerální hustoty kosti je výsledek poslán zpět aplikaci. Implementace matematického modelu, který se nachází na výpočtovém serveru je popsána v podkapitole 4.5.2. Výsledkem výpočtu minerální hustoty kosti je pole hodnot hustoty, které následně vložíme do středů tetrahedrů dané kosti. Poté síti dané kosti nastavíme aktivní skalární pole těchto hodnot. Tento proces se tedy opakuje pro každou jednotlivou zaškrtnutou kost.

```
def draw_realisation(event):
    """ Metoda, která se stará o vykreslení kosti. Nastavuje
    viditlnost aktérům. Metodu recv_array odešle na výpočetní
    server uživatelem nastavené parametry. Výsledná minerální
```

```
    hustota se vloží do vtk sítí a synchonizuje se vtk panel. """
    li.start_loading_spinner(model.vtkpan)
    btn_start_simulation.disabled = True
    set_visibility_actor()
    for bone_type in bone_chcecksbox.value:
        dd = dict(
        sex = sex_buttons.value,
        bone_type = bone_type,
        age = age_slider.value,
        spb = KL_slider.value
        )
        bmd = recv_array(dd)
        model.meshes_rainbow[bone_type].cell_data['BMD [g/cc]'] = bmd
        model.meshes_rainbow[bone_type].set_active_scalars('BMD [g/cc]')
        model.bmd_fun.vector().set_local(bmd)
    model.vtkpan.reset_camera()
    model.vtkpan.synchronize()
    li.stop_loading_spinner(model.vtkpan)
    btn_start_simulation.disabled = False
    file_download.disabled=False
```

Nakonec se resetuje kamera ve vtk panelu a dále se vtk panel synchronizuje, aby se aktualizovalo zobrazení okna na webové stránce. Po celou dobu provádění výpočetního procesu je tlačítko *Start simulation* neaktivní. Tlačítko se znovu zaktivní po dokončení výpočetního procesu.

Obrázek 4.6: Tlačítko Start simulation

4.5.1 Komunikace mezi aplikací a výpočtovým serverem

Aplikace s výpočtovým serverem komunikuje pomocí knihovny *ZeroMQ*. Ve funkci *draw_realisation* se nachází metoda *recv_array*, která se o tuto komunikaci stará. Na straně klienta (aplikace) je vytvořen kontext třídou *Context*, který umožňuje vytvoření soketu typu *REQ*. Poté se soket připojí k výpočtovému serveru a odešle

získané parametry nastavené uživatelem pro výpočtový model kosti k serializaci jako zprávu za pomoci JSON. Potom co se dotaz zpracuje a provede se výpočet minerální hustoty kosti na straně serveru, klient přijme výpočtový model kosti k serializaci opět jako zprávu za pomoci JSON.

```
def recv_array(model_parameters, flags=0, copy=True, track=False):
    """ Vytvoření kontextu a soketu typu REQ. Připojení soketu REQ
    k portu. Poslání parametrů modelu jako \emph{numpy} pole serveru a
    přijmutí metadat(dtype,shape), výpočtového modelu kosti jako
    \emph{numpy} pole. Následná rekonstrukce pole."""
    context = zmq.Context()
    socket = context.socket(zmq.REQ)
    socket.connect("tcp://localhost:5555")
    socket.send_json(model_parameters)
    md = socket.recv_json(flags=flags)
    msg = socket.recv(flags=flags, copy=copy, track=track)
    A = np.frombuffer(msg, dtype=md['dtype'])
    return A.reshape(md['shape'])
```

Numpy pole je typická datová struktura v Pythonu. PyZMQ (ZeroMQ pro python) podporuje odesílání *numpy* polí bez kopírování jakýchkoli dat, protože poskytují rozhraní vyrovnávací paměti Pythonu. Samotná vyrovnávací paměť však nestačí k rekonstrukci pole na přijímací straně. Jsou nutná data dtype a shape. [11]

Na straně výpočtového serveru je také vytvořen kontext. Dále je vytvořen soket typu *REP* a funkcí *bind* se povolí peerům, aby se připojili. Když je ze strany klienta odeslán dotaz, tak ho soket přijme k serializaci opět jako zprávu za pomoci json. Přijatá data se rozdělí a předají matematickému modelu. Nakonec se výsledek výpočtu matematického modelu odešle zpátky klientovi.

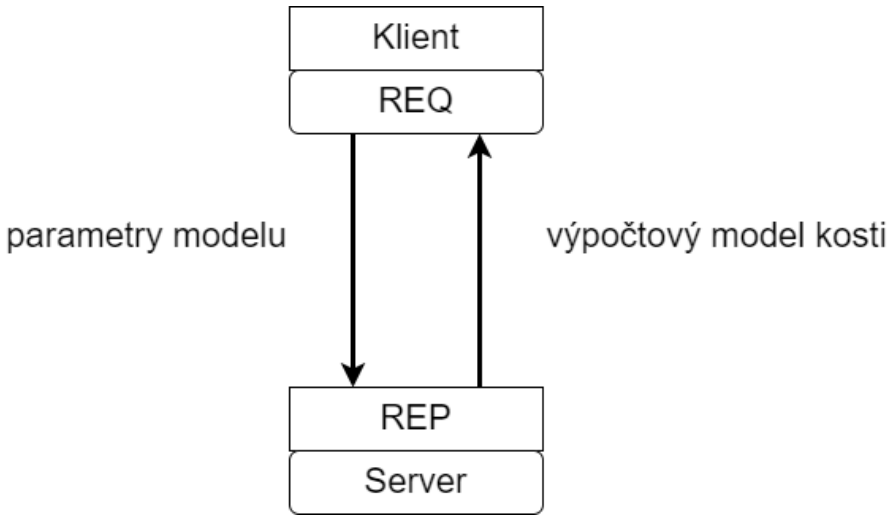
```
def create_socket():
    """Vytvoření kontextu a soketu typu REP. Otevření soketu
    (spojení s portem)"""
    context = zmq.Context()
    socket = context.socket(zmq.REP)
    socket.bind("tcp://*:5555")
    return socket

def send_array(socket, A, flags=0, copy=True, track=False):
    """Poslání metadat a výpočtového modelu kosti jako numpy pole
    zpátky klientovi"""
    md = dict(
        dtype = str(A.dtype),
        shape = A.shape,
    )
```

```
    socket.send_json(md, flags|zmq.SNDMORE)
    return socket.send(A, flags, copy=copy, track=track)

def start_server():
    """Spuštění serveru. Rozdělení a předání dat matematickému modelu."""
    model = KLE()
    logging.info("Data loaded")
    socket = create_socket()
    while True:
        flags=0
        mpr= socket.recv_json(flags|zmq.SNDMORE)
        model.set_sex_and_bone_type(mpr["sex"],mpr["bone_type"])
        model.set_spectral_band(mpr["spb"])
        model.set_age(mpr["age"])
        send_array(socket,model.compute_realisation())
```

Komunikace mezi aplikací a výpočtovým serverem je založena na typu komunikace *request-replay* neboli přeloženo do češtiny žádost-odpověď. Aplikace se serverem komunikuje ve smyčce jak je znázorněno na obrázku 4.7.



Obrázek 4.7: Komunikace mezi aplikací a výpočtovým serverem

4.5.2 Popis implementace matematického modelu

Implementace matematického modelu se nachází na výpočtovém serveru. Jako první se nastaví parametry modelu, které server přijal od klienta.

```
def set_sex_and_bone_type(self, sex, bone_type):
""" Nastavení pohlaví a typu kosti matematickému modelu """
    if sex=='woman':
        self._eigs = self.bmd_eigs_women[bone_type]
        self._vecs = self.bmd_vecs_women[bone_type]
        self._std = self.bmd_std_women[bone_type]
        self._mean = self.bmd_mean_women[bone_type]
        self._slope = self.bmd_slopes_women[bone_type]
        self._sex = 'woman'
    elif sex=='man':
                    ...

def set_spectral_band(self, spb):
""" Nastavení přesnosti matematického modelu
(nastavení vlastních vektorů) """
    lt, ut = spb
    self._thr = slice(lt, ut)

def set_age(self,age):
    """ Nastavení věku pro matematický model """
    self.age = age
```

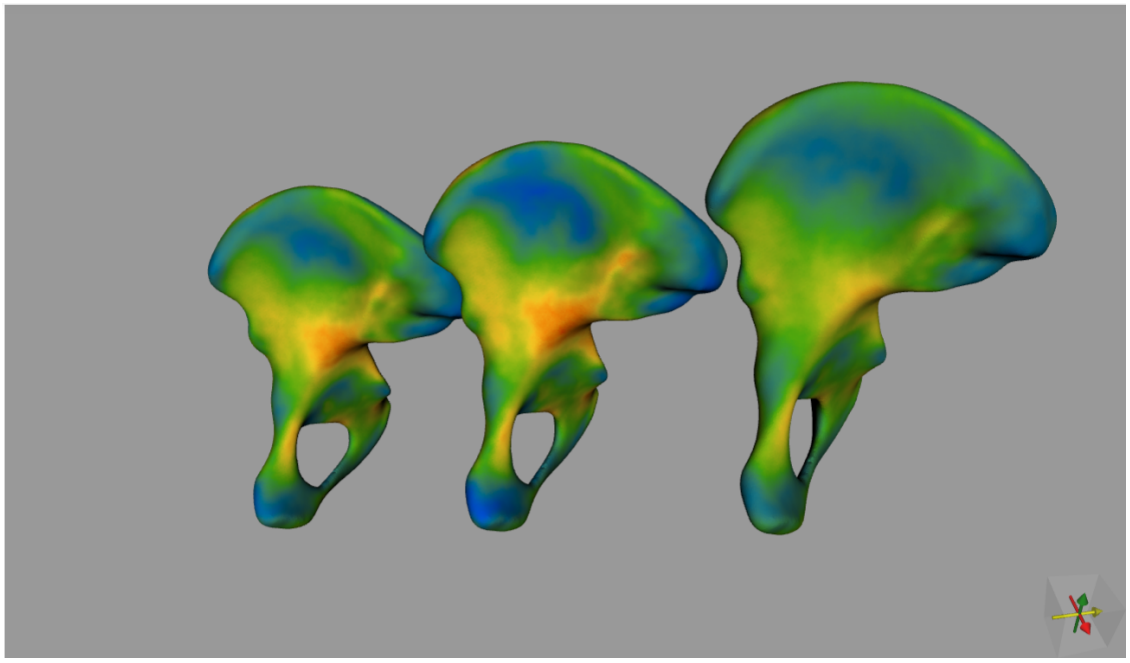
Poté se provede samotný výpočet minerální hustoty kosti pomocí metody *compute_realisation.lhd* a *xi* dohromady vygenerují náhodná čísla z normální P rozměrné distribuce. P odpovídá počtu vybraných vektorů v KL expanzi. Z tohoto vznikne P rozměrné *numpy* pole. bmd_z vychází z rovnice 1.1. Pro každou realizaci se vygenerují nová náhodná čísla.

```
def compute_realisation(self):
    lhd = lhs(n=self._thr.stop - self._thr.start, samples=1)[0]
    xi = stats.norm().ppf(lhd)
    assert (self._thr.stop - self._thr.start) == len(xi)
    bmd_z= np.matmul(np.matmul(self._vecs[:, self._thr],
        self._eigs[self._thr, self._thr]),
        np.diag(xi)).sum(1)
    return self._mean + self._slope * self.age + bmd_z * self._std
```

4.6 Renderovací okno

Objekty které jsou přidané do plotteru jsou v renderovacím okně zobrazeny pomocí panelu vtk. Po spuštění aplikace nejsou v renderovacím okně viditelné žádné objekty. Renderovací okno se jeví uživateli jako prázdné. Až poté co uživatel nastaví parametry a dokončí se výpočetní proces jsou v renderovacím okně zobrazeny kosti s minerální hustotou. Model kosti je 3D objekt a v renderovacím okně s ním může být různě manipulováno. Může se s ním otáčet, naklánět, přibližovat ho nebo vzdalovat. Renderování objektů probíhá přímo v prohlížeči díky knihovně *VTK*.

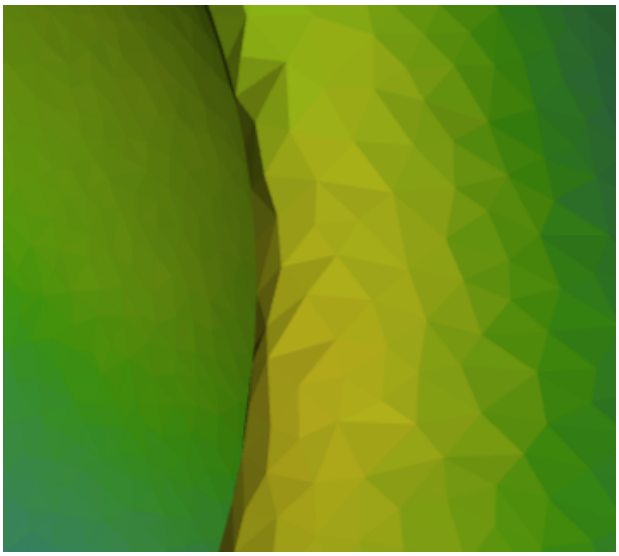
Pokud je zvolen jen jeden typ kosti, kost se vykreslí samotná uprostřed renderovacího okna. Pokud je zvoleno více typů kostí, kosti se vykreslí v řadě vedle sebe. Renderovací okno s více vykreslenými kostmi je vidět na obrázku 4.8. V pravém dolním rohu jsou zobrazeny orientační osy, aby měl uživatel neustále přehled jak je kost zrovna nakloněná. Na orientační widget lze kliknout a otočit scénu ve směru jedné osy.



Obrázek 4.8: Renderovací okno s více modely kostí

Bohužel datasety s ostatními kostmi nebyly v době vytváření aplikace ještě hotové. Proto je na obrázku výše zobrazena levá pánev třikrát. Jakmile budou datasety hotové, budou se takto vedle sebe zobrazovat různé typy kostí. Z implementačního hlediska to nebude představovat problém, jen se nahradí stávající dataset za jiný s novými kostmi.

Minerální hustota je na kosti v renderovacím okně reprezentována prostorovou sítí tetrahedronů. Tetrahedron je složený z bodů, který dále obsahuje souřadnice o tom, jak je propojen s okolními tetrahedrony. Každý tetrahedron v sobě také uchovává hodnotu, což je právě hodnota pro minerální hustotu. Při dostatečném přiblížení kosti lze jednotlivé tetrahedrony dokonce vidět. Detail přiblížené kosti je vidět na obrázku 4.9.



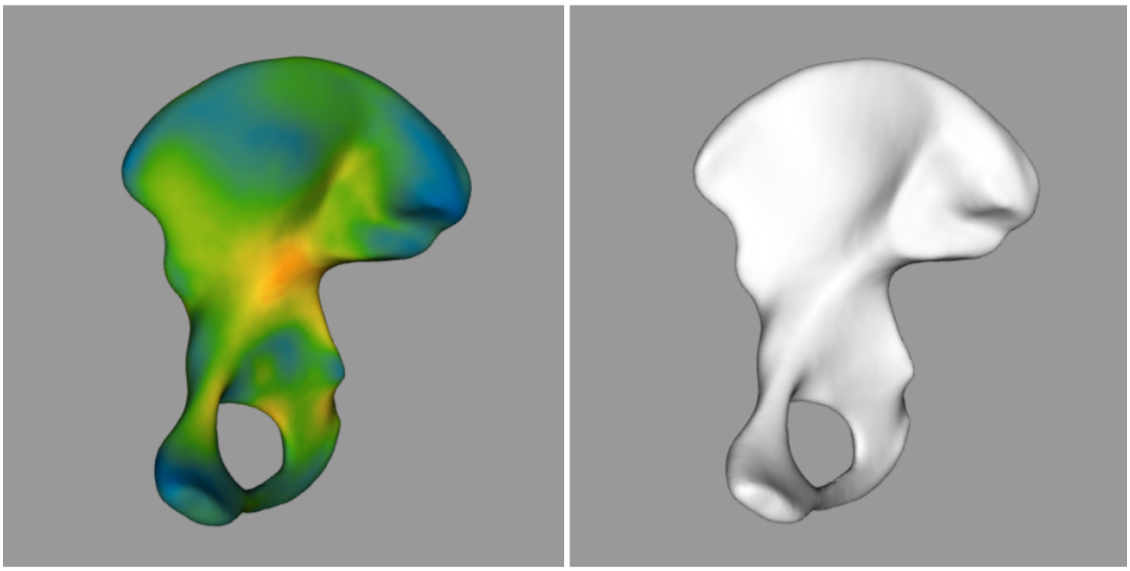
Obrázek 4.9: Síť tetrahedronů

V levé části pod renderovacím oknem jsou vypsány klávesové zkratky, které lze použít pro interakci s ním. Díky těmto zkratkám se výrazně usnadní práce s objekty, které se nachází v renderovacím okně. Aby tyto zkratky fungovaly, kurzor myši se musí nacházet právě v tomto okně. Klávesové zkratky, které je možné použít jsou:

- s – nastaví reprezentaci všech aktérů na *povrch*
- w – nastaví reprezentaci všech aktérů na *drátovou síť*
- v – nastaví reprezentaci všech aktérů na *vrcholy*
- r – vycentruje aktéry a nastaví kameru tak, aby všichni aktéři byli viditelní

V pravé části pod rederovacím oknem se nachází widget, který zobrazí možnosti a tlačítko pro výběr barevné mapy. Tento výběr je vidět na obrázku 4.10. Uživatel má na výběr ze dvou možností. Barevnou mapu *rainbow* využijí spíše inženýři, zatímco barevnou mapu *Greys* zvolí spíše doktoři. Barevná mapa ovlivňuje v jakém barevném rozpoložení se objekty budou zobrazovat. Na obrázku 4.11 je vidět, jak se kost zobrazuje v barevné mapě *rainbow* (vlevo) a barevné mapě *Greys* (vpravo).

Obrázek 4.10: Výběr barevné mapy



Obrázek 4.11: Zobrazení kostí v barevné mapě *rainbow* a *Greys*

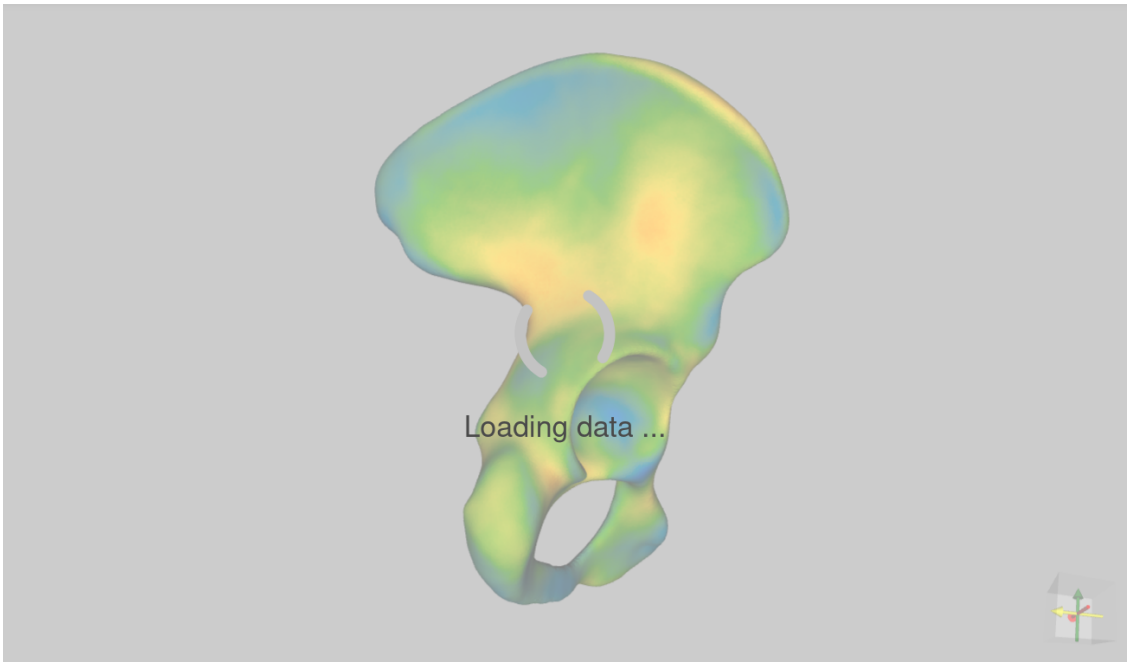
4.7 Loading spinner

Aplikace zobrazí uživateli loading spinner pokaždé, když probíhá načítací, synchronizační nebo výpočetní proces. Tyto procesy trvají kratší i delší chvíli, a proto je nezbytné, aby byl uživatel neustále informován o tom, že se daný proces stále provádí a aplikace nezamrzla.

K vytvoření loading spinneru je použit modul *panel.io.loading*. Tento modul umožňuje, aby se libovolná komponenta *Panel* jevila jako by se načítala a zároveň byla deaktivovaná. Tohoto vzhledu se dosáhne pomocí následujících změn:

- přidání malého spinneru navrch
- zešedivění panelu
- deaktivace panelu
- změna kurzoru myši na spinner při najetí myši na panel

Tento loading spinner se vždy zobrazí v renderovacím okně, jak je vidět na obrázku 4.12. Aby loading spinner odpovídal velikosti renderovacího okna a jeho součástí byl i textový řetězec, musela se jeho třída upravit v css.



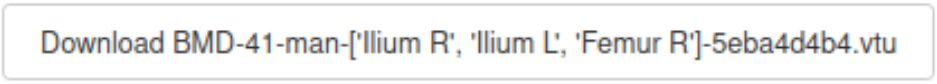
Obrázek 4.12: Loading spinner

4.8 Uložení výpočtového modelu kosti

Pro uložení výpočtového modelu kosti se nachází v postranním panelu tlačítko, které je vidět na obrázku 4.13. Obsah tlačítka tvoří text, který je zároveň názvem ukládaného souboru. Tento text se tvoří dynamicky a mění se v závislosti na tom, jaké parametry uživatel zadává. Název souboru je ve tvaru:

- BMD-věk-pohlaví-[vybrané typy kostí]-unikátní hash.vtu

Unikátní hash se tvoří za pomoci knihovny *uuid*. Slouží k tomu, aby uživatel mohl stahovat více realizací a aby byly od sebe rozlišitelné.

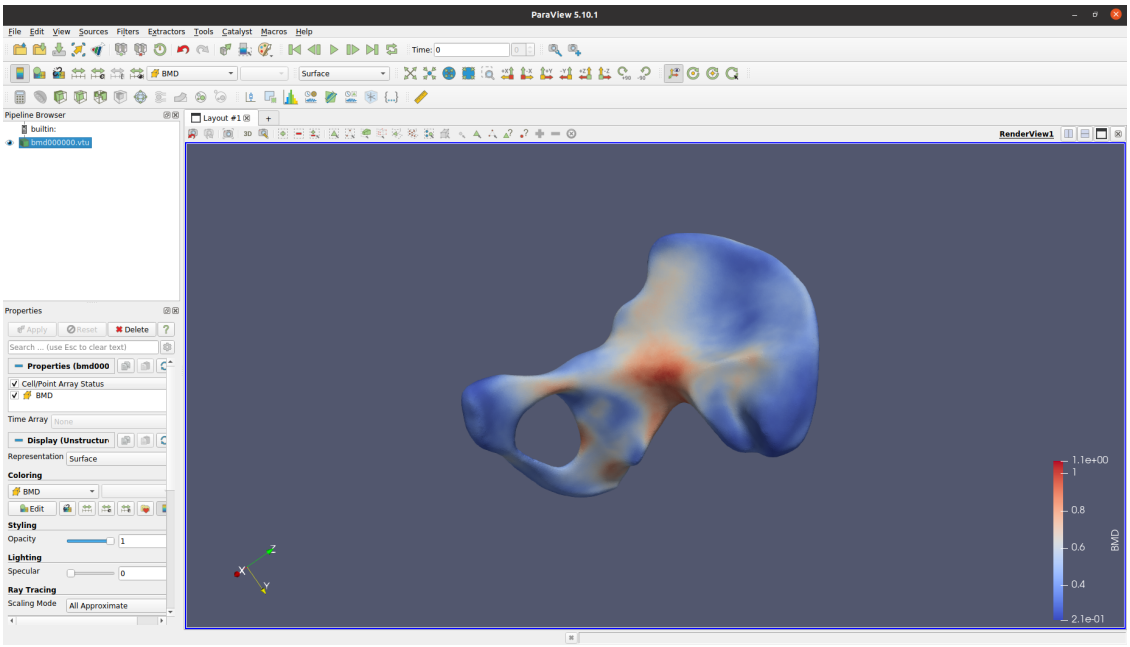


Obrázek 4.13: Tlačítko Download

Tlačítko je po spuštění aplikace neaktivní. Zaktivuje se teprve až po provedení výpočetního procesu a vykreslení kosti do renderovacího okna.

Model kosti je uložen do formátu vtu, což je formát standardně používaný pro výpočty v biomedicínském inženýrství. Takto uložený model kosti lze otevřít v profesionálnějších programech jako je *ParaView*. V těchto programech se s modelem

kosti dá nadále pracovat. Například lze kost rozříznout a zobrazit hodnoty minerální hustoty kosti i uvnitř kosti samotné. Jak model kosti vypadá v programu *ParaView* je vidět na obrázku 4.14.



Obrázek 4.14: Model kosti v ParaView

5 Ověření funkčnosti aplikace

Funkčnost aplikace je ověřena porovnáním výstupů z aplikace s uvedenou literaturou. [1] Jelikož se chování kostní minerální hustoty liší pro jednotlivá pohlaví, ověřování probíhalo zvlášť jak pro muže tak pro ženy. Nejprve bylo nutné ověřit závislost věku na kostní minerální hustotě a poté porovnat střední hodnoty a odchylky.

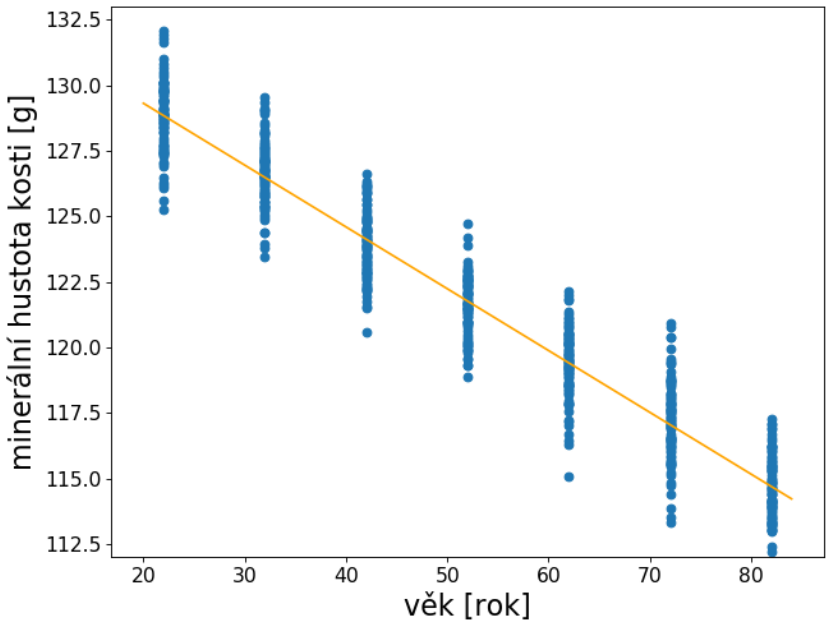
Porovnání je prováděno jen na jednom typu kosti (levá pánev), protože ve studii [1], ze které matematický model vychází, je popsána a analyzována právě jen tato jedna kost.

5.1 Ověření závislosti kostní minerální hustoty na věku

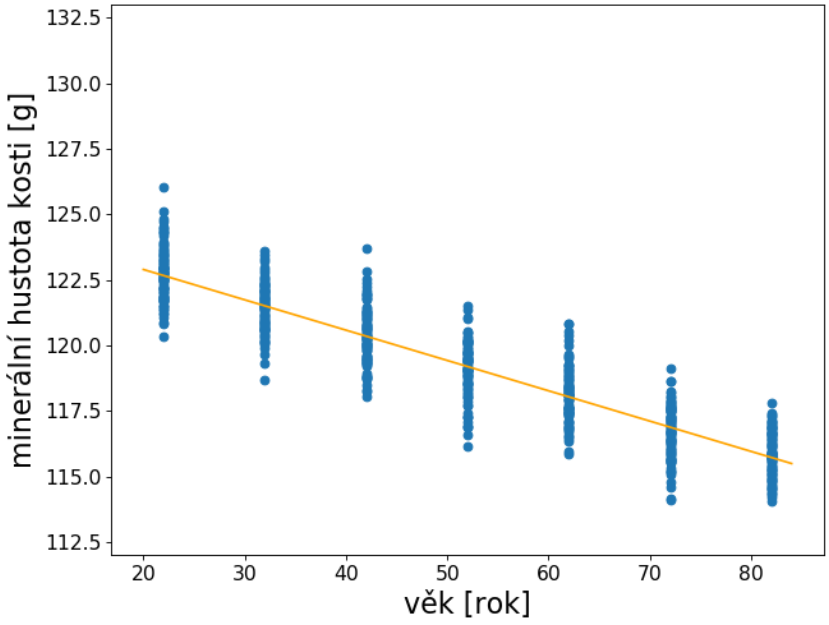
Pro ověření správného vlivu věku bylo nutné vygenerovat 630 realizací výpočtového modelu kosti. Vždy bylo vytvořeno 90 realizací pro každou zvolenou věkovou kategorii (22, 32, 42, 52, 62, 72, 82). Z těchto realizací bylo vytvořeno *numpy* pole, které obsahovalo hmotnost kostní minerální hustoty a věk pro každou realizaci. Tento postup se provedl pro obě pohlaví. Z takto vytvořených dat byly sestaveny grafy pro vizualizaci závislosti kostní minerální hustoty na věku.

Z grafu 5.1 a 5.2 je vidět, že s rostoucím věkem kostní minerální hustota klesá. Také je zde vidět, že u žen je kostní minerální hustota citlivější na věk a tím pádem klesá rychleji než u mužů.

Aplikací vytvořená data a jejich následná vizualizace odpovídá přibližně výsledkům, které jsou uvedeny ve studii. [1]



Obrázek 5.1: Závislost kostní minerální hustoty na věku u žen



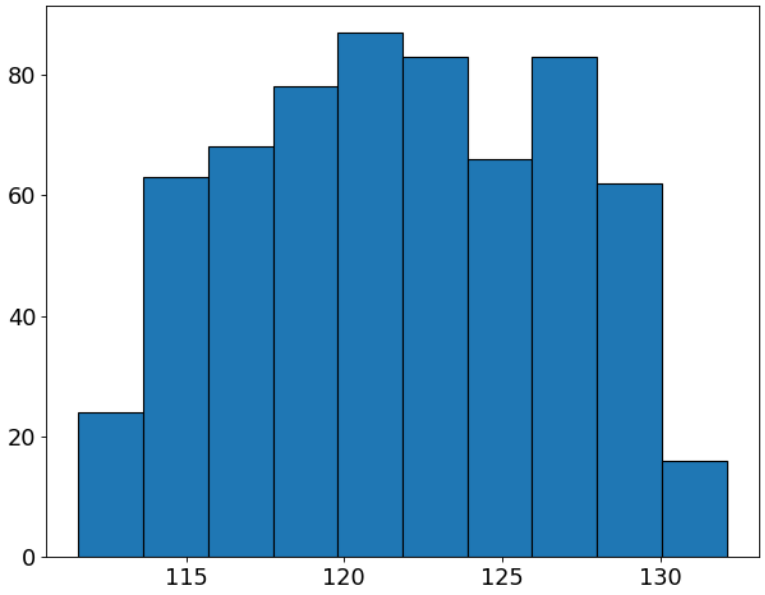
Obrázek 5.2: Závislost kostní minerální hustoty na věku u mužů

5.2 Porovnání střední hodnoty a směrodatné odchylky

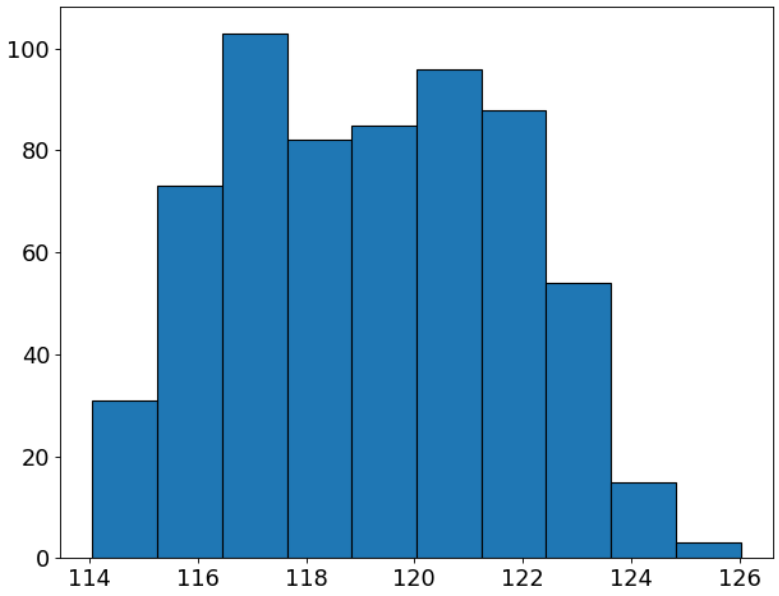
Dále byly vytvořeny histogramy pro důkladnější analýzu. Histogramy byly tvořeny z *numpy* pole obsahující jen hmotnosti minerální hustoty kosti. Na histogramech jsou znázorněny střední hodnoty a směrodatné odchylky. Střední hodnota u žen činila **121,77** g a směrodatná odchylka byla **4,89** g, jak je vidět na histogramu 5.3. U mužů činila střední hodnota **119,20** g a směrodatná odchylka byla **2,53** g, což je vidět na histogramu 5.4.

Studie *„Bone mineral density modeling via random field: Normality, stationarity, sex and age dependence“* [1], ze které vychází matematický model, obsahuje analýzu vzorků dat. Z výsledků v tomto článku plyne, že průměrná hmotnost kosti se pohybuje okolo **120** g.

Z uvedených dat výše lze potvrdit správné fungování aplikace a matematického modelu. Aplikací vytvořené realizace mají střední hodnotu i směrodatnou odchylku v mezích rozsahu, které jsou uvedeny ve studii. [1]



Obrázek 5.3: Histogram kostní minerální hustoty u žen



Obrázek 5.4: Histogram kostní minerální hustoty u mužů

6 Závěr

V rámci této bakalářské práce byla vytvořena interaktivní webová aplikace pro generování realistického modelu kosti. Aplikace byla úspěšně propojena s matematickým modelem a umožňuje vygenerovat výpočtový model kosti dle zadaných parametrů (pohlaví, věk, typ kosti, přesnost). Dále lze tento model kosti uložit do formátu vtu. S modelem kosti v tomto formátu lze nadále pracovat například v programu *ParaView*. Také se podařilo zredukovat čas, kdy by uživatel musel čekat na načtení datasetů. Jelikož jsou datasety poměrně velké, jejich načtení by mohlo trvat i několik minut. Tento problém se podařilo vyřešit pomocí rozdělení aplikace na výpočetní část a uživatelskou část.

Aplikace je přehledná a uživatel má všechny důležité prvky ihned k dispozici, aniž by je musel složitě hledat. Prvky aplikace jsou plně interaktivní a tak se nemůže stát, že by uživatel nevěděl jestli aplikace zamrzla nebo jestli probíhá synchronizační či výpočetní proces.

Aplikace byla ověřena porovnáním výsledků z aplikace s uvedenou literaturou. [1] Bylo potvrzeno, že se správně generuje model kosti se stejnou hmotností, jaká je uvedena ve studii. Také se správně zachycuje vztah mezi hmotností kostní minerální hustoty a věkem pro obě pohlaví. S přibývajícím věkem hmotnost kostní minerální hustoty klesá. Tento pokles je znatelnější u žen. Dále byly ověřeny střední hodnoty a směrodatné odchylky. Jak u mužů tak i u žen odpovídaly hodnoty rozmezím, které jsou uváděny ve studii.

Skvělou a možná i lepší alternativou ke knihovně *Panel*, ve které je aplikace tvořena, je webový framework *trame*. Tento framework také usnadňuje kombinování volně dostupných komponent a vytváření přizpůsobených vizuálních analýz. Zaměřuje se speciálně na *VTK* a ParaView a proto poskytuje kompletní kontrolu nad 3D vizualizacemi a pohybem dat. Tento framework je starý jen pár měsíců. Bohužel v době kdy ho autorka objevila již pracovala s knihovnou *Panel*. [12]

Během tvoření aplikace se objevilo několik problémů na straně knihovny *Panel*. Například se po synchronizaci vtk panelu nezobrazuje scalar bar. Tuto chybu bohužel nejde vyřešit jinak, než ji nahlásit vývojářům *Panel*. Dále základní loading spinner lze zobrazit jen v komponentech typu param. Toto bylo vyřešeno jiným typem loading spinneru. Jeho nevýhodou je, že není plně interaktivní a tak se zobrazuje jen jeden statický text.

Možností dalšího vylepšení aplikace je tu hned několik. Aplikaci by se mohl rozšířit výběr kostí. Nemusí se zaměřovat jen na pánevní kosti, ale klidně i na celou kostru člověka. S tím se pojí možnost, a zároveň potřeba, vylepšit odezvu rychlosti aplikace. V budoucnu by se model kosti mohl doplnit i o nastavení geometrie. Jakožto fyzikální hustota, která je klíčová, tak i tvar kosti je velmi důležitý. Z pohledu implementace by toto rozšíření společně s rozšířením kostí neměl být problém, jelikož je aplikace psána tak, aby se tyto změny mohly jednoduše začlenit. Pokud by se aplikace rozšířila o celou kostru a stala se populární, bude třeba regulovat počet uživatelů.

Literatura

[1] Petr Henyš et al. „Bone mineral density modeling via random field: Normality, stationarity, sex and age dependence“. In: *Computer Methods and Programs in Biomedicine* 210 (2021). ISSN: 0169-2607.

[2] Holoviz contributors. *A high-level app and dashboarding solution for Python [online]*. 2022. URL: https://panel.holoviz.org/ (cit. 09. 10. 2021).

[3] Philipp Rudiger. *Panel Announcement [online]*. 2019. URL: https://blog.holoviz.org/panel_announcement.html (cit. 09. 05. 2022).

[4] HoloViz. *A high-level app and dashboarding solution for Python [online]*. 2022. URL: https://pythonrepo.com/repo/holoviz-panel-python-administrative-interfaces (cit. 09. 05. 2022).

[5] Kitware. *The Visualisation Toolkit (VTK) [online]*. URL: https://vtk.org/ (cit. 09. 10. 2021).

[6] The PyVista Developers. *Pyvista [online]*. 2022. URL: https://docs.pyvista.org/# (cit. 09. 10. 2021).

[7] FEniCS Project. *FEniCSx [online]*. 2021. URL: https://fenicsproject.org/ (cit. 09. 10. 2021).

[8] Bokeh Contributors. *Running a Bokeh server [online]*. 2021. URL: https://docs.bokeh.org/en/2.4.1/docs/user_guide/server.html (cit. 09. 10. 2021).

[9] The ZeroMQ authors. *ZeroMQ [online]*. 2022. URL: https://zeromq.org/get-started/ (cit. 09. 05. 2022).

[10] Freepik. *Anatomy human pelvis on white background Free Vector [online]*. URL: https://www.freepik.com/free-vector/anatomy-human-pelvis-white-background_24466917.htm#query=pelvis&position=34&from_view=keyword (cit. 09. 10. 2021).

[11] Min Ragan-Kelley. *Serializing messages with PyZMQ [online]*. URL: https://het.as.utexas.edu/HET/Software/pyZMQ/serialization.html (cit. 09. 10. 2021).

[12] Kitware contributors. *Trame [online]*. URL: https://github.com/Kitware/trame (cit. 09. 05. 2022).

Přílohy

A Zdrojové kódy

Zdrojové kódy webové aplikace se nachází na Githubu a je možné si je stáhnout z repozitáře. Odkaz na projekt je: https://github.com/HelDoanova/BoneGene_app