Table of Contents

• History of development
• Old architecture
• Modern approach
• New architecture
• Challenges
• Future development
HISTORY OF DEVELOPMENT
History of development

• 29 separate applications
• Shared code
• PHP and MySQL
• Single server
Old architecture - Idea

https://server_domain_name/app_name

APP 1 ➔ APP 2 ➔ APP 3

APP 4 ➔ APP 6

APP 7 ➔ APP 8 ➔ APP 9

DB ➔ Scheduled jobs for data import

SIS ➔ 3rd party services
Old architecture – new requirements

- InfoSec vulnerability concerns
- Data becomes outdated often
- New features/Edit existing ones
- Bug fixes
Old architecture – reality (problems)
MODERN APPROACH
Modern approach

• Needed something anyone on the team can support
• Separation of concerns
• Improve security
• Improve resilience
• Improve data consistency
• Become proactive to problems
Modern approach – development choice

- Change programming language (moved to C#)
- Change database solution (moved to MSSQL)
- Use code-first approach with DB
Modern approach – managed services choice

Use as many managed services as possible:

- Databases -> AWS RDS, AWS ElastiCache
- Servers -> AWS Fargate or AWS Lambda
- Security -> Secrets manager, scanning of containers, API Gateway
- Logging -> AWS CloudWatch
NEW ARCHITECTURE
New architecture
CHALLENGES
Challenges

- Learning curve
- Performance
- No disruption to users
FUTURE DEVELOPMENT
Future development

- Automate the entire deployment with CI/CD
- Move rest of the apps
- Re-evaluate our infrastructure after 12 months (performance, price, etc.)
QUESTIONS?